Module 10
Sequential Decision Making

Dr. Jitesh H. Panchal

Decision Making in Engineering Design

School of Mechanical Engineering
Purdue University, West Lafayette, IN
Module Outline

1. Sequential Decision Making Problems
   - Deterministic Dynamic Problems
   - Stochastic Dynamic Problems

2. The Dynamic Programming Algorithm
   - Dynamic Programming for Deterministic Problems
   - Stochastic Dynamic Programming
   - Challenges in Dynamic Programming

3. Approximate Dynamic Programming
   - Myopic policies
   - Lookahead policies
   - Policy function approximations
   - Value function approximations

---

Bertsekas, D.P., 2019, *Reinforcement Learning and Optimal Control*, Athena Scientific, Nashua, NH, USA.
Sequential Decision Making Problems

Deterministic Dynamic Problems
Stochastic Dynamic Problems
Sequential Decision Making in Design

Sequential Decision Making is an activity of gathering information about alternatives to compare and choose the best alternative.

It consists of sequential decisions to:

- Choose alternative from the decision space
- Select an information source or an evaluation method
- Decide whether to stop gathering information

Example:

[Diagram showing the process of selecting designs, evaluating, deciding to stop, and selecting the best design.]
Example: Engineering Design Optimization

**Decision**: Select shape and geometry of a machine component

Decision-making approaches:

<table>
<thead>
<tr>
<th>Normative</th>
<th>Model uncertainty and maximize the expected performance (utility)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptive</td>
<td>Select an existing design and modify</td>
</tr>
</tbody>
</table>

Figure: Jet Engine Bracket

www.grabcad.com/challenges/ge-jet-engine-bracket-challenge
Example: Motion Planning

**Decision**: Select a sequence of configurations that move the robot from point A to B

Decision-making approaches:

<table>
<thead>
<tr>
<th>Normative</th>
<th>Map the room and optimize the route</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptive</td>
<td>Turn away if you encounter wall or staircase</td>
</tr>
</tbody>
</table>

Figure: Roomba [1]

Example: Information Acquisition Decisions

Consult Economist (EMV=822)

Economist says “Market up” (0.485)

High-Risk Stock (EMV = 580)
Low-Risk Stock (EMV = 540)
Savings account (EMV = 500)

Economist says “Market flat” (0.300)

Economist says “Market Down” (0.215)

Dr. Jitesh H. Panchal
Decision Trees for Modeling Sequential Decisions
A Classification of Sequential Decision Making Problems

1. Deterministic vs. Stochastic problems
2. Discrete vs. Continuous problems
3. Finite horizon vs. Infinite Horizon problems
Deterministic Discrete Problems

Sequential Decision Making

The Dynamic Programming Algorithm
Approximate Dynamic Programming

Deterministic Dynamic Problems
Stochastic Dynamic Problems
Deterministic Discrete Problems

Example: Shortest Path Problem

Figure: A shortest path problem
Deterministic Discrete Problems

\[ V_0(S_0) = \max_{a_0} \{ C_0(S_0, a_0) + V_1(S_1) \} \]

\[ V_1(S_1) = \max_{a_1} \{ C_1(S_1, a_1) + V_2(S_2) \} \]
Deterministic Discrete Problems

Bellman’s equation:

\[ V_t(S_t) = \max_{a_t} \{ C_t(S_t, a_t) + V_{t+1}(S_{t+1}) \} \quad \forall S_t \in S_t \]

\[ S_{t+1} = S^M(S_t, a_t) \]

where,
- \( a_t \) = action at timestep \( t \)
- \( S \) = set of all possible states
- \( C_t(S_t, a_t) \) = contribution or reward generated from state \( S_t \) and taking the action \( a_t \)
- \( S^M(S_t, a_t) \) = transition function
Deterministic Continuous-space Optimal Control Problem

Initial Temp. $S_0$ → Oven 1 Temperature $a_0$ → $S_1$ → Oven 1 Temperature $a_1$ → Final Temp $S_2$

$$S_{t+1} = S^M(S_t, a_t)$$
$$= (1 - \beta)S_t + \beta a_t \quad 0 < \beta < 1, \quad t = 0, 1$$

Objective: To get the final temperature $S_2$ close to a given target $T$, while expending relatively little energy.

Minimize:
- Terminal cost $= r(S_2 - T)^2$
- Total cost of energy $= \sum_t a_t^2$
Sequential Decision Making Problems

The Dynamic Programming Algorithm

Approximate Dynamic Programming

Deterministic Dynamic Problems

Stochastic Dynamic Problems

Stochastic Dynamic Problems

\[ V_t(S_t) = \max_{a_t} \mathbb{E}\{C_t(S_t, a_t) + V_{t+1}(S_{t+1}(S_t, a_t))\} \quad \forall S_t \in S_t \]
Continuous Stochastic Dynamic Problems

Initial Temp. $S_0$ → Oven 1 Temperature $a_0$ → Oven 1 Temperature $a_1$ → Final Temp. $S_2$

$$S_{t+1} = S^M(S_t, a_t, W_t)$$
$$= (1 - \beta)S_t + \beta a_t + W_t$$
$$0 < \beta < 1, \quad t = 0, 1$$

$W_0$ and $W_1$ are independent random variables with given distribution, and zero mean: $\mathbb{E}[W_0] = \mathbb{E}[W_1] = 0$
Sequential Decision Making Problems
The Dynamic Programming Algorithm
Approximate Dynamic Programming
Dynamic Programming for Deterministic Problems
Stochastic Dynamic Programming
Challenges in Dynamic Programming
The tail of an optimal sequence is optimal for the tail subproblem.
Let \( \{a_0^*, a_1^*, \ldots, a_{N-1}^*\} \) be an optimal sequence of actions, which together with \( S_{t+1} = S^M(S_t, a_t) \) and an initial state \( S_0 \) determines the corresponding state sequence \( \{S_1^*, S_2^*, \ldots, S_N^*\} \).

Consider the subproblem whereby we start at \( S_k^* \) at time \( k \) and wish to maximize the value from time \( k \) to time \( N \) over \( \{a_k, a_{k+1}, \ldots, a_{N-1}\} \). Then, the truncated optimal actions \( \{a_k^*, a_{k+1}^*, \ldots, a_{N-1}^*\} \) is optimal for this subproblem.
The Dynamic Programming (DP) algorithm proceeds sequentially, by solving all the tail sub-problems of a given time length, using the solution of the tail subproblems of shorter time length.
The Dynamic Programming Algorithm

Example: Deterministic Shortest Path Problem

Figure: A shortest path problem
The Dynamic Programming Algorithm

Example: Deterministic Shortest Path Problem
Example: Linear Quadratic Optimal Control Problem

End $t = 2$: $V_2(S_2) = r(S_2 - T)^2$

State transition: $S_2 = (1 - \beta)S_1 + \beta a_1$

For $t = 1$: $V_1(S_1) = \min_{a_1}[a_1^2 + V_2(S_2)]$
Example: Linear Quadratic Optimal Control Problem

Initial Temp. $S_0$ → Oven 1 Temperature $a_0$ → $S_1$ → Oven 1 Temperature $a_1$ → Final Temp $S_2$
Example: Linear Quadratic Optimal Control Problem

For \( t = 0 \): 

\[
V_0(S_0) = \min_{a_0} [a_0^2 + V_1(S_1)]
\]

where 

\[
V_1(S_1) = \frac{r[(1 - \beta)^2 S_0 + \beta(1 - \beta)a_0 - T]^2}{(1 + r\beta^2)}
\]
Example: Linear Quadratic Optimal Control Problem

**Optimal Policy**

\[
\begin{align*}
\text{Optimal policy:} \\
a_0^*(S_0) &= \frac{r\beta(1 - \beta)(T - (1 - \beta)^2S_0)}{1 + r\beta^2(1 + (1 - \beta)^2)} \\
a_1^*(S_1) &= \frac{r\beta(T - (1 - \beta)S_1)}{1 + r\beta^2}
\end{align*}
\]

**Optimal Cost:**

\[
\begin{align*}
V_0(S_0) &= \frac{r[(1 - \beta)^2S_0 - T]^2}{1 + r\beta^2(1 + (1 - \beta)^2)}
\end{align*}
\]
Comparison of Stochastic and Deterministic Programming Algorithms

Differences between Stochastic and Deterministic Programming

- Use of expected values/utilities
- Policies vs. Control sequences

A policy is a rule (or function) that determines a decision given the available information in state $S_t$. 
Stochastic Linear Quadratic Problem

\[
S_{t+1} = S^M(S_t, a_t, W_t) \\
= (1 - \beta)S_t + \beta a_t + W_t \\
0 < \beta < 1
\]

where, \( t = 0, 1 \)
\( W_0 \) and \( W_1 \) are independent random variables with given distribution, and zero mean: \( \mathbb{E}[W_0] = \mathbb{E}[W_1] = 0 \)
Stochastic Linear Quadratic Problem

For $t = 2$:  
$$V_2(S_2) = r(S_2 - T)^2$$

State transition:
$$S_2 = (1 - \beta)S_1 + \beta a_1 + W_1$$

For $t = 1$,
$$V_1(S_1) = \min_{a_1} \mathbb{E}[a_1^2 + V_2(S_2)]$$
$$= \min_{a_1} \mathbb{E}[a_1^2 + r(S_2 - T)^2]$$
$$= \min_{a_1} \mathbb{E}[a_1^2 + r((1 - \beta)S_1 + \beta a_1 + W_1 - T)^2]$$
State transition:

\[ S_1 = (1 - \beta)S_0 + \beta a_0 + W_0 \]

For \( t = 0 \),

\[ V_0(S_0) = \min_{a_0} \mathbb{E}[a_0^2 + V_1(S_1)] \]
If $C(S_t, a_t)$ is known with certainty, then

$$V_t(S_t) = \max_{a_t} \mathbb{E}\{C_t(S_t, a_t) + V_{t+1}(S_{t+1}(S_t, a_t))\}$$

$$= \max_{a_t}(C_t(S_t, a_t) + \mathbb{E} V_{t+1}(S_{t+1}(S_t, a_t)))$$

$$= \max_{a_t} \left(C_t(S_t, a_t) + \sum_{s' \in S} P(s' | S_t, a_t) V_{t+1}(s') \right)$$
Curses of Dimensionality:

1. **State space**: If the state variable, $S_t = (S_{t1}, S_{t2}, \ldots, S_{tI})$ has $I$ dimensions, and if $S_{ti}$ can take up to $L$ possible values, then we might have up to $L^I$ different states.

2. **Outcome space**: The random variable $W_t = (W_{t1}, W_{t2}, \ldots, W_{tJ})$ might have $J$ dimensions. If $W_{tj}$ can take on $M$ outcomes, then our outcome space might take on up to $M^J$ outcomes.

3. **Action space**: The action vector $x_t = (x_{t1}, x_{t2}, \ldots, x_{tK})$ might have $K$ dimensions. If $x_{tk}$ can take on $N$ outcomes, then we might have up to $N^K$ outcomes.

---

Approximate Dynamic Programming
Approximate Dynamic Programming: Strategies

1. Myopic policies
2. Lookahead policies
3. Policy function approximations
4. Value function approximations

Myopic policies optimize costs/rewards now, but do not explicitly use forecasted information or any direct representation of decisions in the future.

\[ A^{\text{Myopic}}(S_t) = \arg \max_a C(S_t, a) \]
2. Lookahead policies

Lookahead policies explicitly optimize over some horizon by combining:

- an approximation of future information, and
- an approximation of future actions.

Look ahead policies explicitly represent future information and decisions.

Lookahead strategies can be computationally expensive.
Examples of Lookahead policies

1. **Tree search**: enumerate all possible actions and all possible outcomes over $T$ time periods (solving a decision tree).

2. **Sparse sampling tree search**: replace the enumeration over all outcomes with a Monte Carlo sample of outcomes.

3. **Rollout heuristics**: replace the explicit enumeration of the entire tree with some heuristic policy (e.g., myopic policy) to evaluate what might happen after reaching a state.

4. **Rolling horizon procedures**: Rolling the horizon, one period at a time.
3. Policy function approximations

Policy function approximations are functions that directly return an action given a state, without resorting to any form of imbedded optimization, and without directly using any forecast of future information.

Three types of policy function approximations:

1. Lookup tables
2. Parametric representations
3. Nonparametric representations
4. Value function approximations

Value function approximation based policies depend on an approximation of the value $V_t(S_t)$ as a result of a decision made now.

The impact of a decision now on the future is captured purely through a value function that depends on the state that results from a decision now.

Examples:

$$\tilde{V}^n(S^n) = (1 - \alpha)\tilde{V}^{n-1}(S^n) + \alpha \hat{V}^n$$

$$\tilde{V}(S) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S)$$
**Approaches for Approximation in Value Space**

\[ V_t(S_t) = \max_{a_t} \mathbb{E}\{C_t(S_t, a_t) + V_{t+1}(S_{t+1}(S_t, a_t))\} \]

1. **Approximate \( \mathbb{E}\{\cdot\} \)**
   - Certainty equivalence
   - Adaptive simulation
   - Monte Carlo tree search

2. **Approximate \( V_{t+1}(S_{t+1}(S_t, a_t)) \)**
   - Problem approximation
   - Rollout
   - Parametric approximation
   - Neural Nets

3. **Approximate \( \max_{a_t}\{\cdot\} \)**
   - Discretization
Approximate Dynamic Programming: Sample Algorithm

Let $\bar{V}_t(S_t) \approx V_t(S_t)$. Update the approximation iteratively.

For the $n^{th}$ sample:

$$\hat{v}_t^n = \max_{a_t} \left( C_t(S^n_t, a_t) + \sum_{s' \in S} P(s' | S^n_t, a_t) \bar{V}^{n-1}_{t+1}(s') \right)$$

Note that $\bar{V}^{n-1}_{t+1}$ is computed using information from iterations $n = 1, \ldots, (n-1)$.

Update value function: $\bar{V}_t^n(S_t) = \begin{cases} \hat{v}_t^n, & S_t = S^n_t, \\ \bar{V}^{n-1}_t(S_t), & \text{otherwise} \end{cases}$
Approximate Dynamic Programming: Q-Learning Algorithm

\( Q(s, a) = \) the value of being in state \( s \) and taking an action \( a \).

\[
V(s) = \max_a Q(s, a)
\]

Let \( \bar{Q}^n(s, a) \) be an estimate of the true \( Q(s, a) \) after \( n \) iterations. Choose an action \( a^n = \arg \max_a \bar{Q}^{n-1}(S^n, a) \)

After choosing \( a^n \), observe contribution \( \hat{C}(S^n, a^n) \) and the next state \( S^{n+1} \). Update \( \bar{Q} \) as follows:

\[
\hat{q}^n = \hat{C}(S^n, a^n) + \max_{a'} \bar{Q}^{n-1}(S^{n+1}, a')
\]

\[
\bar{Q}_t^n(S^n, a^n) = (1 - \alpha_{n-1})\bar{Q}^{n-1}(S^n, a^n) + \alpha_{n-1}\hat{q}^n
\]
1 Sequential Decision Making Problems
   - Deterministic Dynamic Problems
   - Stochastic Dynamic Problems

2 The Dynamic Programming Algorithm
   - Dynamic Programming for Deterministic Problems
   - Stochastic Dynamic Programming
   - Challenges in Dynamic Programming

3 Approximate Dynamic Programming
   - Myopic policies
   - Lookahead policies
   - Policy function approximations
   - Value function approximations
References
