ECE276B: Planning & Learning in Robotics
Lecture 4: The Dynamic Programming Algorithm

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistant:

Thai Duong: tduong@eng.ucsd.edu

UC San Diego

JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Dynamic Programming

- **Control policy**: a function π that maps a time step $t \in \mathbb{N}$ and a state $\mathbf{x} \in \mathcal{X}$ to a feasible control input $\mathbf{u} \in \mathcal{U}(\mathbf{x})$
- **Value function** $V_t^\pi(\mathbf{x})$: estimates how good in terms of expected long-term cost/reward it is to be in state $\mathbf{x}$ at time t and follow policy π
- **Objective**: construct an optimal control policy:

$$\pi^* = \arg\min_\pi V_0^\pi(\mathbf{x}_0)$$

- **Dynamic Programming Algorithm** can compute an optimal control policy given a known MDP model of the environment
  - Idea: uses the value function to structure the search for good policies
  - Generality: can handle non-convex and non-linear problems
  - Complexity: polynomial in the number of states and actions
  - Efficiency: much more efficient than the brute-force approach of evaluating all possible strategies.

Principle of Optimality

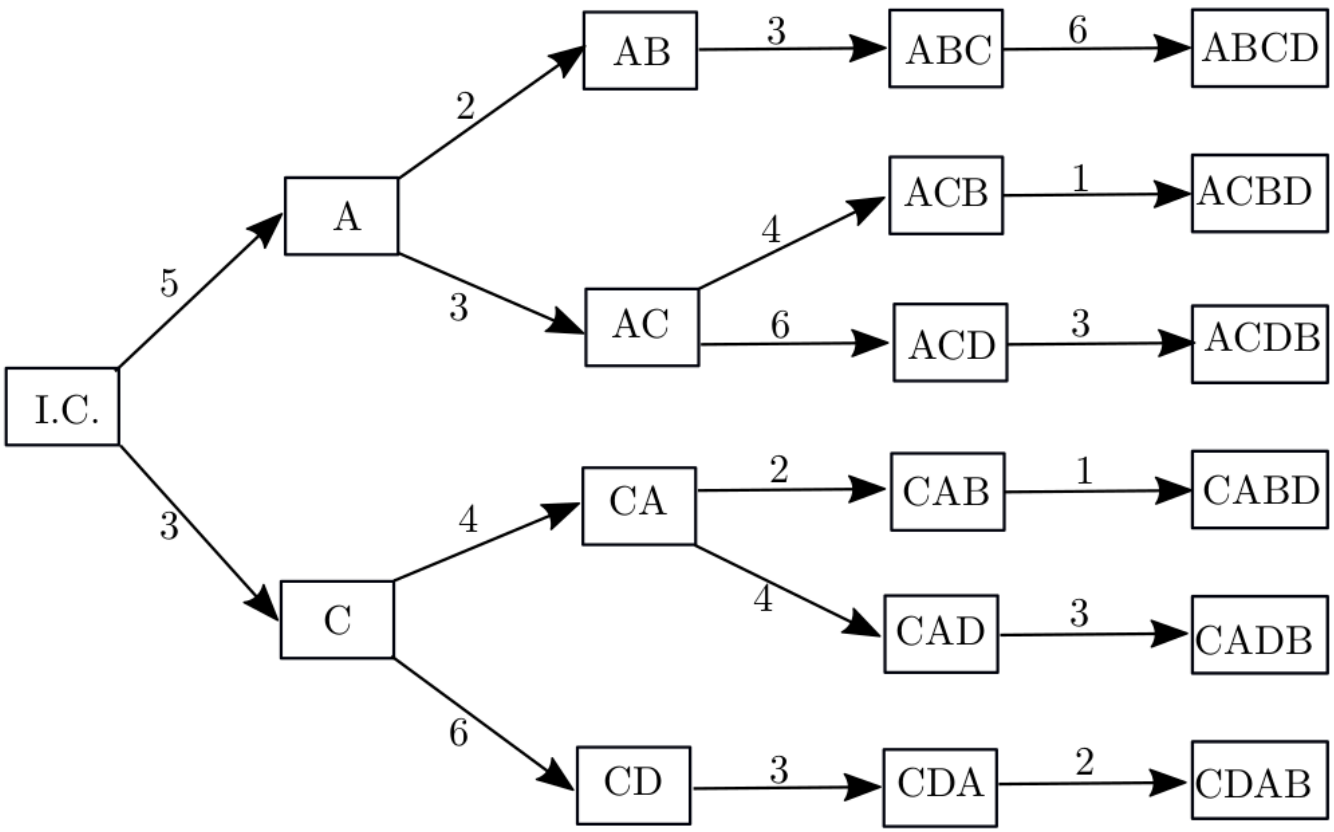
- Let $\pi^*_{0:T-1}$ be an optimal control policy
- Consider a **subproblem**, minimizing the value at state $\mathbf{x}$ at time t:

$$V_t^\pi(\mathbf{x}) = \mathbb{E}_{\mathbf{x}_{t+1:T}}\left[\gamma^{T-t} q(\mathbf{x}_T) + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau)) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right]$$

- **Principle of optimality**: the truncated policy $\pi^*_{t:T-1}$ is optimal for the subproblem starting at time t
- **Intuition**: Suppose $\pi^*_{t:T-1}$ were not optimal for the subproblem. Then, there would exist a policy yielding a lower cost on at least some portion of the state space.

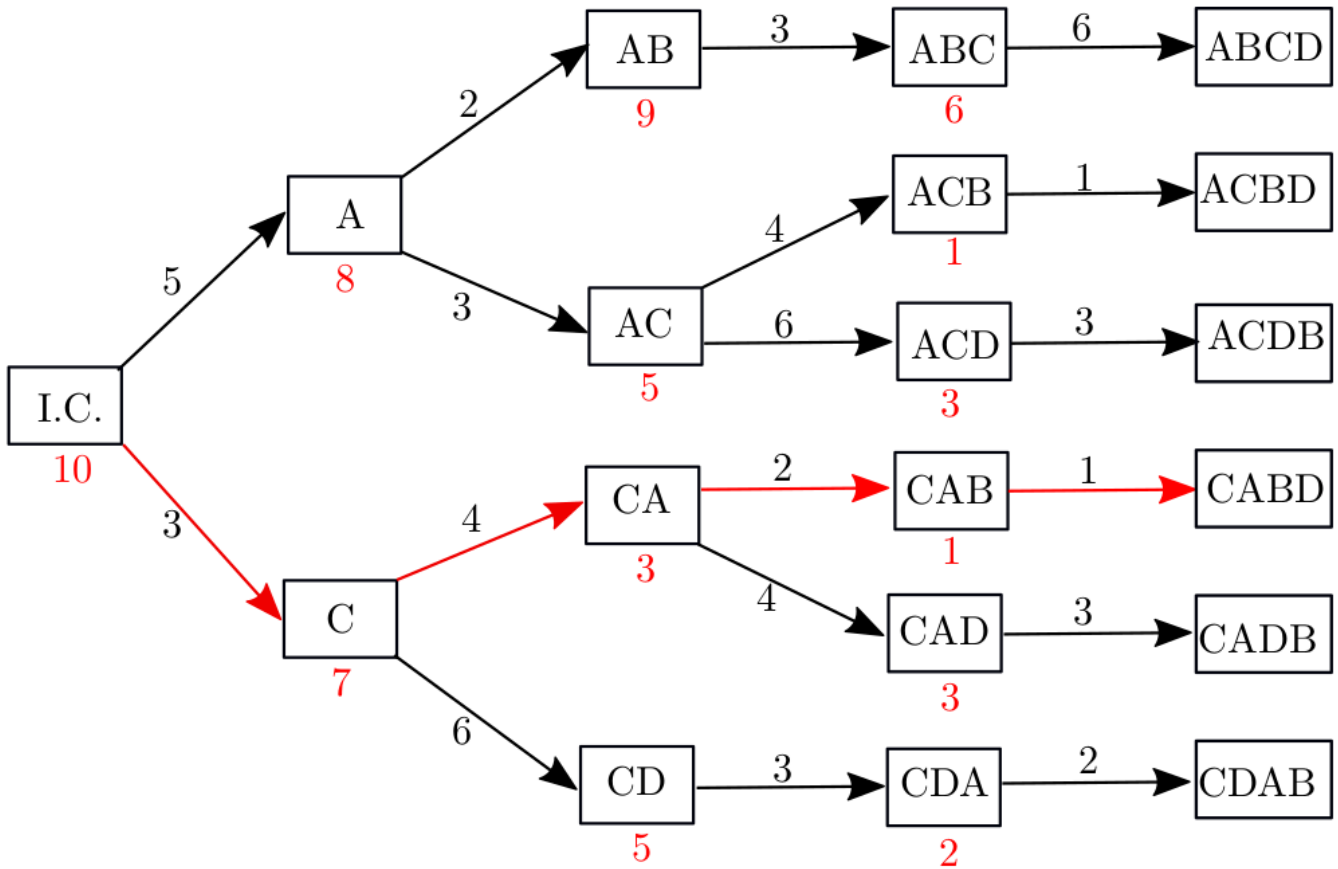
Example: Deterministic Scheduling Problem

- Consider a deterministic scheduling problem where 4 operations A, B, C, D are used to produce a product
- Rules: Operation A must occur before B, and C before D
- Cost: there is a transition cost between each two operations:



Example: Deterministic Scheduling Problem

- Dynamic programming is applied backwards in time. First, construct an optimal solution at the last stage and then work backwards.
- The optimal cost-to-go at each state of the scheduling problem is denoted with red text below the state:



The Dynamic Programming Algorithm

Algorithm 1 Dynamic Programming

Input: MDP $(\mathcal{X}, \mathcal{U}, p_0, p_f, T, \ell, \mathfrak{q}, \gamma)$

$V_T(\mathbf{x}) = \mathfrak{q}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}$

for $t = (T-1) \dots 0$ **do**

$Q_t(\mathbf{x}, \mathbf{u}) = \ell(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}_{\mathbf{x}' \sim p_f(\cdot|\mathbf{x},\mathbf{u})} \left[V_{t+1}(\mathbf{x}') \right], \quad \forall \mathbf{x} \in \mathcal{X}, \mathbf{u} \in \mathcal{U}(\mathbf{x})$

$V_t(\mathbf{x}) = \min_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} Q_t(\mathbf{x}, \mathbf{u}), \qquad \forall \mathbf{x} \in \mathcal{X}$

$\pi_t(\mathbf{x}) = \arg\min_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} Q_t(\mathbf{x}, \mathbf{u}), \qquad \forall \mathbf{x} \in \mathcal{X}$

return policy $\pi_{0:T-1}$ and value function V_0

The expected value function at $\mathbf{x}' \sim p_f(\cdot|\mathbf{x}, \mathbf{u})$ is:

- Continuous $\mathcal{X}$: $\mathbb{E}_{\mathbf{x}' \sim p_f(\cdot|\mathbf{x},\mathbf{u})} \left[V_{t+1}(\mathbf{x}') \right] = \int V_{t+1}(\mathbf{x}') p_f(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) d\mathbf{x}'$
- Discrete $\mathcal{X}$: $\mathbb{E}_{\mathbf{x}' \sim p_f(\cdot|\mathbf{x},\mathbf{u})} \left[V_{t+1}(\mathbf{x}') \right] = \sum_{\mathbf{x}' \in \mathcal{X}} V_{t+1}(\mathbf{x}') p_f(\mathbf{x}' \mid \mathbf{x}, \mathbf{u})$

The Dynamic Programming Algorithm

- At each step, all possible states $\mathbf{x} \in \mathcal{X}$ are considered because we do not know a priori which states will be visited
- This point-wise optimization at each $\mathbf{x} \in \mathcal{X}$ is what gives us a policy $\pi_t(\mathbf{x})$, i.e., a function specifying a control input for **every** state $\mathbf{x} \in \mathcal{X}$
- Consider a discrete-space example with $|\mathcal{X}| = 10$ states, $|\mathcal{U}| = 10$ control inputs, planning horizon $T = 4$, and given x_0:
  - There are $|\mathcal{U}|^T = 10^4$ different open-loop strategies
  - There are $|\mathcal{U}|^{|\mathcal{X}|(T-1)+1} = 10^{31}$ different closed-loop strategies
  - For each stage t and each state $\mathbf{x}$, the DP algorithm goes through the $|\mathcal{U}|$ control inputs to determine the optimal input. In total, there are $|\mathcal{U}||\mathcal{X}|(T-1) + |\mathcal{U}| = 310$ such operations.

Dynamic Programming Optimality

Theorem: Optimality of the DP Algorithm

The policy $\pi_{0:T-1}$ and value function V_0 returned by the DP algorithm are optimal for the finite-horizon optimal control problem.

- **Proof**:
  - Let $V_t^*(\mathbf{x})$ be the optimal cost for the $(T-t)$-stage problem that starts at time t in state $\mathbf{x}$.
  - Proceed by induction
  - **Base-case**: $V_T^*(\mathbf{x}) = q(\mathbf{x}) = V_T(\mathbf{x})$
  - **Hypothesis**: Assume that for $t+1$, $V_{t+1}^*(\mathbf{x}) = V_{t+1}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$
  - **Induction**: Show that $V_t^*(\mathbf{x}) = V_t(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$

Proof of Dynamic Programming Optimality

$$
\begin{aligned}
V_t^*(\mathbf{x}_t) &= \min_{\pi_{t:T-1}} \mathbb{E}_{\mathbf{x}_{t+1:T}|\mathbf{x}_t} \left[\gamma^{T-t} q(\mathbf{x}_T) + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau)) \right] \\
&= \min_{\pi_{t:T-1}} \mathbb{E}_{\mathbf{x}_{t+1:T}|\mathbf{x}_t} \left[\ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t)) + \gamma^{T-t} q(\mathbf{x}_T) + \sum_{\tau=t+1}^{T-1} \gamma^{\tau-t} \ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau)) \right] \\
&\overset{(1)}{=\!=} \min_{\pi_{t:T-1}} \ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t)) + \mathbb{E}_{\mathbf{x}_{t+1:T}|\mathbf{x}_t} \left[\gamma^{T-t} q(\mathbf{x}_T) + \sum_{\tau=t+1}^{T-1} \gamma^{\tau-t} \ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau)) \right] \\
&\overset{(2)}{=\!=} \min_{\pi_{t:T-1}} \ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t)) + \gamma \mathbb{E}_{\mathbf{x}_{t+1}|\mathbf{x}_t} \left[\mathbb{E}_{\mathbf{x}_{t+2:T}|\mathbf{x}_{t+1}} \left[\gamma^{T-t-1} q(\mathbf{x}_T) + \sum_{\tau=t+1}^{T-1} \gamma^{\tau-t-1} \ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau)) \right] \right] \\
&\overset{(3)}{=\!=} \min_{\pi_t} \left\{ \ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t)) + \gamma \mathbb{E}_{\mathbf{x}_{t+1}|\mathbf{x}_t} \left[\min_{\pi_{t+1:T-1}} \mathbb{E}_{\mathbf{x}_{t+2:T}|\mathbf{x}_{t+1}} \left[\gamma^{T-t-1} q(\mathbf{x}_T) + \sum_{\tau=t+1}^{T-1} \gamma^{\tau-t-1} \ell(\mathbf{x}_\tau, \pi_\tau(\mathbf{x}_\tau)) \right] \right] \right\} \\
&\overset{(4)}{=\!=} \min_{\pi_t} \left\{ \ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t)) + \gamma \mathbb{E}_{\mathbf{x}_{t+1} \sim p_f(\cdot|\mathbf{x}_t, \pi_t(\mathbf{x}_t))} \left[V_{t+1}^*(\mathbf{x}_{t+1}) \right] \right\} \\
&\overset{(5)}{=\!=} \min_{\mathbf{u}_t \in \mathcal{U}(\mathbf{x}_t)} \left\{ \ell(\mathbf{x}_t, \mathbf{u}_t) + \gamma \mathbb{E}_{\mathbf{x}_{t+1} \sim p_f(\cdot|\mathbf{x}_t, \mathbf{u}_t)} \left[V_{t+1}(\mathbf{x}_{t+1}) \right] \right\} \\
&= V_t(\mathbf{x}_t), \quad \forall \mathbf{x}_t \in \mathcal{X}
\end{aligned}
$$

Proof of Dynamic Programming Optimality

(1) Since $\ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t))$ is not a function of $\mathbf{x}_{t+1:T}$

(2) Using conditional probability
$p(\mathbf{x}_{t+1:T}|\mathbf{x}_t) = p(\mathbf{x}_{t+2:T}|\mathbf{x}_{t+1}, \mathbf{x}_t)p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ and the Markov assumption

(3) The minimization can be split since the term $\ell(\mathbf{x}_t, \pi_t(\mathbf{x}_t))$ does not depend on $\pi_{t+1:T-1}$. The expectation $\mathbb{E}_{\mathbf{x}_{t+1}|\mathbf{x}_t}$ and $\min_{\pi_{t+1:T}}$ can be exchanged since the functions $\pi_{t+1:T-1}$ make the cost small for all initial conditions., i.e., independently of $\mathbf{x}_{t+1}$.

- (1)-(3) is the *principle of optimality*

(4) By definition of $V^*_{t+1}(\cdot)$ and the motion model $\mathbf{x}_{t+1} \sim p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t)$

(5) By the induction hypothesis

Example: Chess Strategy Optimization

- State: $x_t \in \mathcal{X} := \{-2, -1, 0, 1, 2\}$ – the difference between our and the opponent's score at the end of game t
- Input: $u_t \in \mathcal{U} := \{timid, bold\}$
- Dynamics: with $p_d > p_w$:

$$\begin{aligned} p_f(x_{t+1} = x_t \mid u_t = timid, x_t) &= p_d \\ p_f(x_{t+1} = x_t - 1 \mid u_t = timid, x_t) &= 1 - p_d \\ p_f(x_{t+1} = x_t + 1 \mid u_t = bold, x_t) &= p_w \\ p_f(x_{t+1} = x_t - 1 \mid u_t = bold, x_t) &= 1 - p_w \end{aligned}$$

- Cost: $V_t(x_t) = \mathbb{E}\left[\mathfrak{q}(x_2) + \sum_{\tau=t}^{1} \underbrace{\ell(x_\tau, u_\tau)}_{=0}\right]$ with

$$\mathfrak{q}(x) = \begin{cases} -1 & \text{if } x > 0 \\ -p_w & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Dynamic Programming Applied to the Chess Problem

- Initialize: $V_2(x_2) = \begin{cases} -1 & \text{if } x_2 > 0 \\ -p_w & \text{if } x_2 = 0 \\ 0 & \text{if } x_2 < 0 \end{cases}$
- Recursion: for all $x_t \in \mathcal{X}$ and $t = 1, 0$:

$$V_t(x_t) = \min_{u_t \in \mathcal{U}} \left\{ \ell(x_t, u_t) + \mathbb{E}_{x_{t+1}|x_t,u_t} \left[V_{t+1}(x_{t+1}) \right] \right\}$$

$$= \min \left\{ \underbrace{p_d V_{t+1}(x_t) + (1 - p_d) V_{t+1}(x_t - 1)}_{\text{timid}}, \underbrace{p_w V_{t+1}(x_t + 1) + (1 - p_w) V_{t+1}(x_t - 1)}_{\text{bold}} \right\}$$

Dynamic Programming Applied to the Chess Problem

- $x_1 = 1$:

$$V_1(1) = -\max\{p_d + (1-p_d)p_w, p_w + (1-p_w)p_w\} \underset{p_d > p_w}{\overset{\text{since}}{=\!=}}$$
$$= -p_d - (1-p_d)p_w$$
$$\pi_1^*(1) = \textit{timid}$$

- $x_1 = 0$:

$$V_1(0) = -\max\{p_d p_w + (1-p_d)0, p_w + (1-p_w)0\} = -p_w$$
$$\pi_1^*(0) = \textit{bold}$$

- $x_1 = -1$:

$$V_1(-1) = -\max\{p_d 0 + (1-p_d)0, p_w p_w + (1-p_w)0\} = -p_w^2$$
$$\pi_1^*(-1) = \textit{bold}$$

Dynamic Programming Applied to the Chess Problem

- $x_0 = 0$:

$$
\begin{aligned}
V_0(0) &= -\max\{p_d V_1(0) + (1-p_d)V_1(-1), p_w V_1(1) + (1-p_w)V_1(-1)\} \\
&= -\max\left\{p_d p_w + (1-p_d)p_w^2, p_w(p_d + (1-p_d)p_w) + (1-p_w)p_w^2\right\} \\
&= -p_d p_w - (1-p_d)p_w^2 - (1-p_w)p_w^2
\end{aligned}
$$

$\pi_0^*(0) = \textit{bold}$

- As before, the optimal strategy is to play timid iff ahead in the score.