# Reinforcement Learning and Optimal Control

## ASU, CSE 691, Winter 2019

Dimitri P. Bertsekas
dimitrib@mit.edu

Lecture 2

# Outline

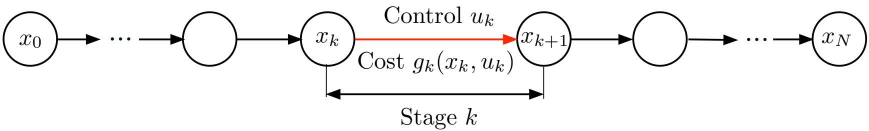
1. Review of Exact Deterministic DP Algorithm

2. Examples: Discrete/Combinatorial DP Problems

3. Stochastic DP Algorithm

4. Problem Formulations and Simplifications

# Finite Horizon Deterministic Problem




- System

$$x_{k+1} = f_k(x_k, u_k), \qquad k = 0, 1, \ldots, N-1$$

where $x_k$: State, $u_k$: Control chosen from some set $U_k(x_k)$

- Cost function:

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

- For given initial state $x_0$, minimize over control sequences $\{u_0, \ldots, u_{N-1}\}$

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

- Optimal cost function $J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0,\ldots,N-1}} J(x_0; u_0, \ldots, u_{N-1})$

# DP Algorithm: Solving Progressively Longer Tail Subproblems

## Go backward to compute the optimal costs $J_k^*(x_k)$ of the $x_k$-tail subproblems

Start with

$$J_N^*(x_N) = g_N(x_N), \qquad \text{for all } x_N,$$

and for $k = 0, \ldots, N-1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \Big[ g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big], \qquad \text{for all } x_k.$$

Then optimal cost $J^*(x_0)$ is obtained at the last step: $J_0^*(x_0) = J^*(x_0)$.

## Go forward to construct optimal control sequence $\{u_0^*, \ldots, u_{N-1}^*\}$

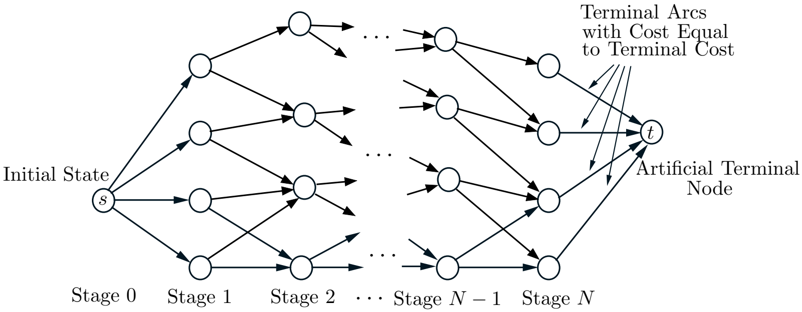
Start with

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} \Big[ g_0(x_0, u_0) + J_1^*\big(f_0(x_0, u_0)\big) \Big], \qquad x_1^* = f_0(x_0, u_0^*).$$

Sequentially, going forward, for $k = 1, 2, \ldots, N-1$, set

$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} \Big[ g_k(x_k^*, u_k) + J_{k+1}^*\big(f_k(x_k^*, u_k)\big) \Big], \qquad x_{k+1}^* = f_k(x_k^*, u_k^*).$$

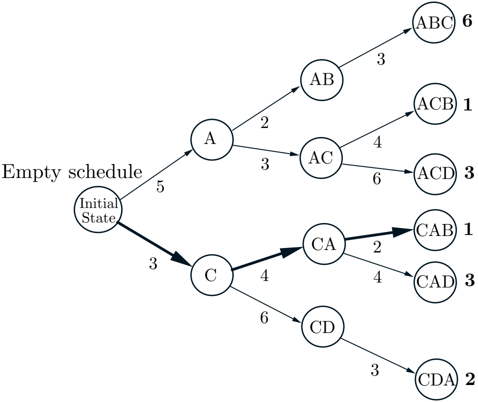
Interesting fact for the future: We can replace $J_k^*$ with an approximation $\tilde{J}_k$.




- Nodes correspond to states $x_k$
- Arcs correspond to state-control pairs $(x_k, u_k)$
- An arc $(x_k, u_k)$ has start and end nodes $x_k$ and $x_{k+1} = f_k(x_k, u_k)$
- An arc $(x_k, u_k)$ has a cost $g_k(x_k, u_k)$. The cost to optimize is the sum of the arc costs from the initial node $s$ to the terminal node $t$.
- The problem is equivalent to finding a minimum cost/shortest path from $s$ to $t$.

Interesting fact for the future: There are several alternative (exact and approximate) shortest path algorithms.

# Discrete-State Deterministic Scheduling Example

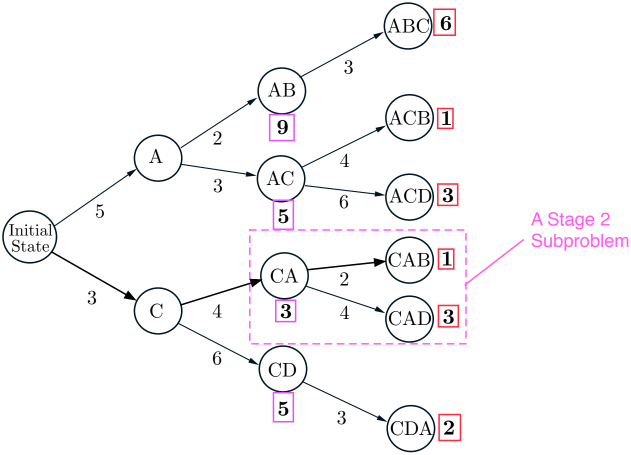



Find optimal sequence of operations A, B, C, D (A must precede B and C must precede D)

## DP Problem Formulation

- States: Partial schedules; Controls: Stage 0, 1, and 2 decisions; Cost data shown along the arcs
- Recall the DP idea: Break down the problem into smaller pieces (tail subproblems)
- Start from the last decision and go backwards

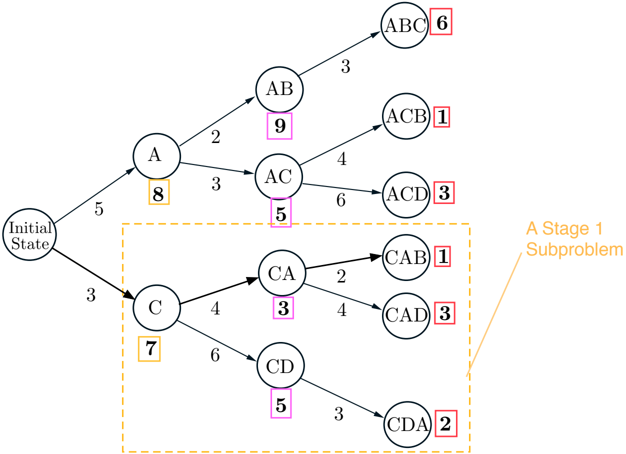
# DP Algorithm: Stage 2 Tail Subproblems




**Solve the stage 2 subproblems (using the terminal costs - in red)**

At each state of stage 2, we record the optimal cost-to-go and the optimal decision

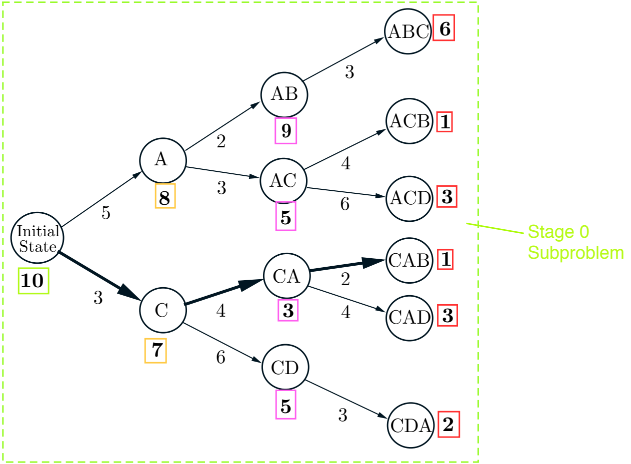
# DP Algorithm: Stage 1 Tail Subproblems




Solve the stage 1 subproblems (using the optimal costs of stage 2 subproblems - in purple)

At each state of stage 1, we record the optimal cost-to-go and the optimal decision

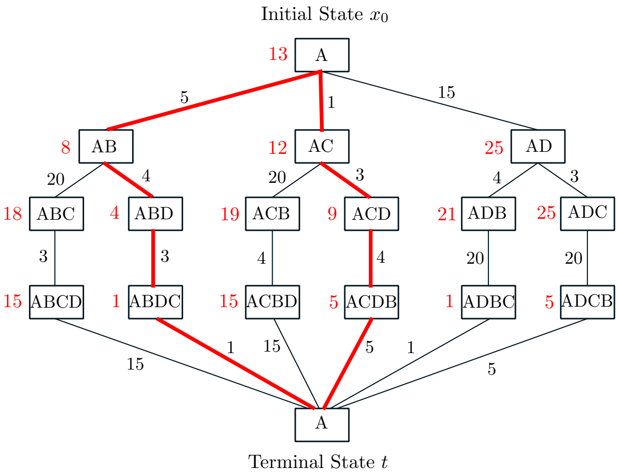
# DP Algorithm: Stage 0 Tail Subproblems




**Solve the stage 0 subproblem (using the optimal costs of stage 1 subproblems - in orange)**

- The stage 0 subproblem is the entire problem
- The optimal value of the stage 0 subproblem is the optimal cost $J^*$(initial state)

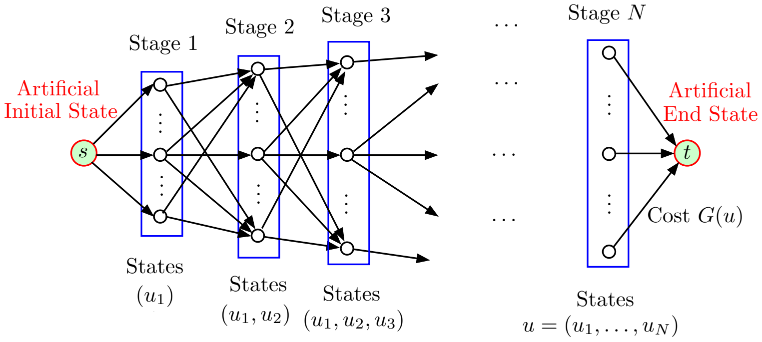
# Combinatorial Optimization: Traveling Salesman Example




Matrix of Intercity Travel Costs

| | | | |
|---|---|---|---|
| | 5 | 1 | 15 |
| 5 | | 20 | 4 |
| 1 | 20 | | **3** |
| 15 | 4 | 3 | |

# General Discrete Optimization

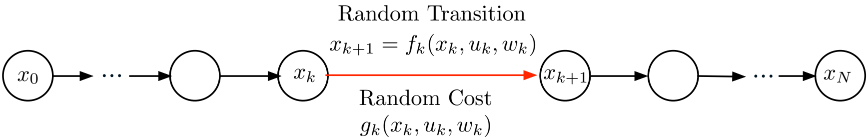



## Minimize $G(u)$ subject to $u \in U$

- Assume that each solution $u$ has $N$ components: $u = (u_1, \ldots, u_N)$
- View the components as the controls of $N$ stages
- Define $x_k = (u_1, \ldots, u_k)$, $k = 1, \ldots, N$, and introduce artificial states $x_0$ and $x_N$
- Define just terminal cost as $G(u)$; all other costs are 0

This formulation often makes little sense for exact DP, but a lot of sense for approximate DP/approximation in value space

# Stochastic DP Problems




- System $x_{k+1} = f_k(x_k, u_k, w_k)$ with random "disturbance" $w_k$ (e.g., physical noise, market uncertainties, demand for inventory, unpredictable breakdowns, etc)
- Cost function:

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

- Policies $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where $\mu_k$ is a "closed-loop control law" or "feedback policy"/a function of $x_k$. Specifies control $u_k = \mu_k(x_k)$ to apply when at $x_k$.
- For given initial state $x_0$, minimize over all $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ the cost

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

- Optimal cost function $J^*(x_0) = \min_\pi J_\pi(x_0)$

# The Stochastic DP Algorithm

## Produces the optimal costs $J_k^*(x_k)$ of the tail subproblems that start at $x_k$

Start with $J_N^*(x_N) = g_N(x_N)$, and for $k = 0, \ldots, N-1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \Big\}, \qquad \text{for all } x_k.$$

- The optimal cost $J^*(x_0)$ is obtained at the last step: $J_0^*(x_0) = J^*(x_0)$.
- The optimal control function $\mu_k^*$ is constructed simultaneously with $J_k^*$, and consists of the minimizing $u_k^* = \mu_k^*(x_k)$ above.

## Online implementation of the optimal policy, given $J_1^*, \ldots, J_{N-1}^*$

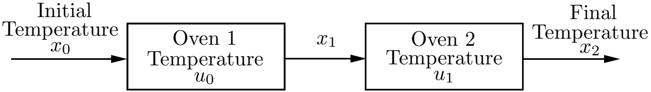
Sequentially, going forward, for $k = 0, 1, \ldots, N-1$, observe $x_k$ and apply

$$u_k^* \in \arg \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \Big\}.$$

Issues: Need to compute $J_{k+1}^*$ (possibly off-line), compute expectation for each $u_k$, minimize over all $u_k$

Approximation in value space: Use $\tilde{J}_k$ in place of $J_k^*$; approximate $E\{\cdot\}$ and $\min_{u_k}$.

# Linear Quadratic Problem




- System: $x_{k+1} = (1-a)x_k + au_k + w_k$ ($w_k$ is random and 0-mean)
- Cost: $E\{r(x_N - T)^2 + \sum_{k=0}^{N-1} u_k^2\}$
- DP algorithm for $N = 2$

$$J_2^*(x_2) = r(x_2 - T)^2,$$

$$J_1^*(x_1) = \min_{u_1} E_{x_2}\{u_1^2 + J_2^*(x_2)\} = \min_{u_1} E_{w_1}\left\{u_1^2 + r\big((1-a)x_1 + au_1 + w_1 - T\big)^2\right\}$$

To obtain optimal $\mu_1^*(x_1)$, set $\nabla_{u_1} J_1^* = 0$, use $E\{w_1\} = 0$, and solve:

$$\mu_1^*(x_1) = \frac{raT}{1+ra^2} - \frac{ra(1-a)x_1}{1+ra^2} \quad \text{(linear in } x_1\text{)}$$

Plug into the expression for $J_1^*$, to obtain

$$J_1^*(x_1) = \frac{r\big((1-a)x_1 - T\big)^2}{1+ra^2} + rE\{w_1^2\}$$

# Linear Quadratic Problem (Continued) - Certainty Equivalence

- The stage 1 DP calculation gives a form of $J_1^*$ that is similar to the one for $J_2^*$:

$$J_1^*(x_1) = \frac{r\big((1-a)x_1 - T\big)^2}{1+ra^2} + rE\{w_1^2\}$$

- We plug the expression for $J_1^*$ into the DP equation for $J_0^*$:

$$J_0^*(x_0) = \min_{u_0} E_{w_0}\left\{u_0^2 + \frac{r\big((1-a)((1-a)x_0 + au_0 + w_0) - T\big)^2}{1+ra^2}\right\} + rE\{w_1^2\}$$

- To obtain optimal $\mu_0^*(x_0)$, set $\nabla_{u_0} J_0^* = 0$, use $E\{w_0\} = 0$, and solve:

$$\mu_0^*(x_0) = \frac{r(1-a)aT}{1+ra^2\big(1+(1-a)^2\big)} - \frac{(1-a)^2 x_0}{1+ra^2\big(1+(1-a)^2\big)}$$

- The result is the same as if $w_1$ and $w_0$ were set to their expected values ($=0$).
- This is called certainty equivalence, and generalizes to more complex types of linear quadratic problems.
- For other problems it may be used as basis for approximation.

# DP Algorithm for Q-Factors

- Optimal Q-factors are given by

$$Q_k^*(x_k, u_k) = E\Big\{g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big)\Big\}$$

They define optimal policies and optimal cost-to-go functions by

$$\mu_k^*(x_k) \in \arg \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k), \qquad J_k^*(x_k) = \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k)$$

- DP algorithm can be written in terms of Q-factors

$$Q_k^*(x_k, u_k) = E\left\{g_k(x_k, u_k, w_k) + \min_{u_{k+1}} Q_{k+1}^*\big(f_k(x_k, u_k, w_k), u_{k+1}\big)\right\}$$

Some math magic: With $E\{\cdot\}$ outside the min, the right side can be approximated by sampling and simulation.

- Approximately optimal Q-factors $\tilde{Q}_k(x_k, u_k)$, define suboptimal policies and suboptimal cost-to-go functions by

$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k) \qquad \tilde{J}_k(x_k) = \min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k)$$

# How do we Formulate DP Problems?

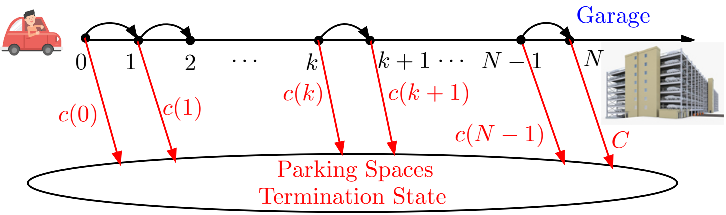
## An informal recipe: First define the stages and then the states

Define as state $x_k$ something that summarizes the past for future optimization purposes, i.e., as long as we know $x_k$, all past information is irrelevant.

## Some examples

- In the traveling salesman problem, we need to include all the info (past cities visited) in the state.
- In the linear quadratic problem, when we select the oven temperature $u_k$, the total info available is everything we have seen so far, i.e., the material and oven temperatures $x_0, u_0, x_1, u_1, \ldots, u_{k-1}, x_k$. However, all the useful information at time $k$ is summarized in just $x_k$.
- In partial or imperfect information problems, we use "noisy" measurements for control of some quantity of interest $y_k$ that evolves over time (e.g., the position/velocity vector of a moving object). If $I_k$ is the collection of all measurements up to time $k$, it is correct to use $I_k$ as state.
- It may also be correct to use alternative states; e.g., the conditional probability distribution $P_k(y_k \mid I_k)$. This is called belief state, and should subsume all the information that is useful for the purposes of control choice.

# Problems with a Terminal State: A Parking Example



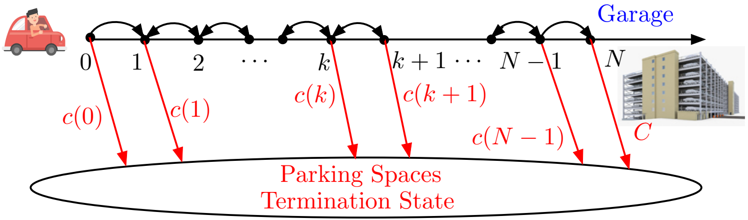

- Start at spot 0; either park at spot $k$ with cost $c(k)$ (if free) or continue; park at garage at cost $C$ if not earlier.
- Spot $k$ is free with a priori probability $p(k)$, and its status is observed upon reaching it.
- How do we formulate the problem as a DP problem?

**We have three states. $F$: current spot is free, $\overline{F}$: current spot is taken, parked state**

$$J^*_{N-1}(F) = \min\big[c(N-1),\ C\big], \qquad J^*_{N-1}(\overline{F}) = C$$

$$J^*_k(F) = \min\Big[c(k),\ p(k+1)J^*_{k+1}(F) + \big(1 - p(k+1)\big)J^*_{k+1}(\overline{F})\Big], \qquad \text{for } k = 0, \ldots, N-2$$

$$J^*_k(\overline{F}) = p(k+1)J^*_{k+1}(F) + \big(1 - p(k+1)\big)J^*_{k+1}(\overline{F}), \qquad \text{for } k = 0, \ldots, N-2$$




- Bidirectional parking: We can go back to parking spots we have visited at a cost
  - “Easy case:" The status of already seen spots stays unchanged
  - “Complex case:" The status of already seen spots changes stochastically
- Correlations of the status of different parking spots
- More complicated parking lot topologies
- Multiagent versions: Multiple drivers and “searchers"
- Our homework will revolve around versions of the parking problem

# About the Next Lecture

**We will cover:**

- General principles of approximation in value and policy space
- Problem approximation methods (enforced decomposition, probabilistic approximation)

PLEASE READ AS MUCH OF SECTIONS 2.1, 2.2 AS YOU CAN