A Series of Lectures on Approximate Dynamic Programming Lecture 1

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

University of Cyprus
September 2017

Our Aim

Discuss optimization by Dynamic Programming (DP)

and the use of approximations

Purpose: Computational tractability in a broad variety of practical contexts

The Scope of these Lectures

After an intoduction to exact DP, we will focus on approximate DP for optimal control under stochastic uncertainty

- The subject is broad with rich variety of theory/math, algorithms, and applications
- Applications come from a vast array of areas: control/robotics/planning, operations research, economics, artificial intelligence, and beyond ...
- We will concentrate on control of discrete-time systems mostly for a finite number of stages (a finite horizon), and the expected value criterion
- We will focus mostly on algorithms ... less on theory and modeling

We will not cover:

- Infinite horizon problems (except briefly)
- Imperfect state information and minimax/game problems
- Simulation-based methods: reinforcement learning, neuro-dynamic programming
- A series of video lectures on the latter can be found at the author's web site

Reference: The lectures will follow Chapters 1 and 6 of the author's book

"Dynamic Programming and Optimal Control," Vol. I, Athena Scientific, 2017

Lectures Plan

Overview of Exact DP

- The basic problem formulation; some examples
- The DP algorithm for finite horizon problems with perfect state information
- Computational limitations; motivation for approximate DP

Approximate DP - I (limited lookahead with cost-to-go approximation)

- Approximation in value space
- Parametric cost approximation, feature-based architectures, neural networks

Approximate DP - II (model-free and simulation-based ideas)

- Q-factor approximation and model-free approximate DP
- Simulation-based on-line approximation; rollout and Monte Carlo tree search
- Applications in backgammon and AlphaGo

Approximate DP - III (some alternative approaches)

- Approximation in policy space
- Limited lookahead with problem approximation

EXACT DYNAMINC PROGRAMMING

Outline

Basic Problem Structure for DP

Discrete-time system

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

- x_k: State; summarizes past information that is relevant for future optimization at time k
- u_k: Control; decision to be selected at time k from a given set $U_k(x_k)$
- w_k: Disturbance; random parameter with distribution $P(w_k \mid x_k, u_k)$
- For deterministic problems there is no w_k

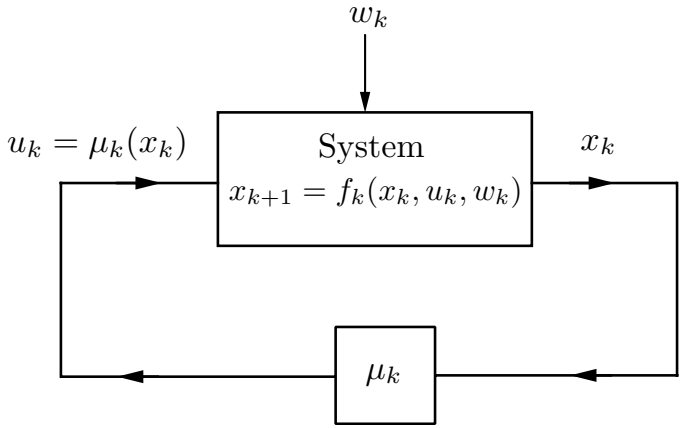
Cost function that is additive over time

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

Perfect state information

The control u_k is applied with (exact) knowledge of the state x_k

Optimization over Feedback Policies



- Feedback policies: Rules that specify the control to apply at each possible state x_k that can occur
- Major distinction: We minimize over sequences of functions of state $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, with $u_k = \mu_k(x_k) \in U_k(x_k)$ - not sequences of controls $\{u_0, u_1, \ldots, u_{N-1}\}$

Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ starting at initial state x_0

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

Optimal cost function:

$$J^*(x_0) = \min_\pi J_\pi(x_0)$$

Scope of DP

Any optimization (deterministic, stochastic, minimax, etc) involving a sequence of decisions fits the framework

A continuous-state example: Linear-quadratic optimal control

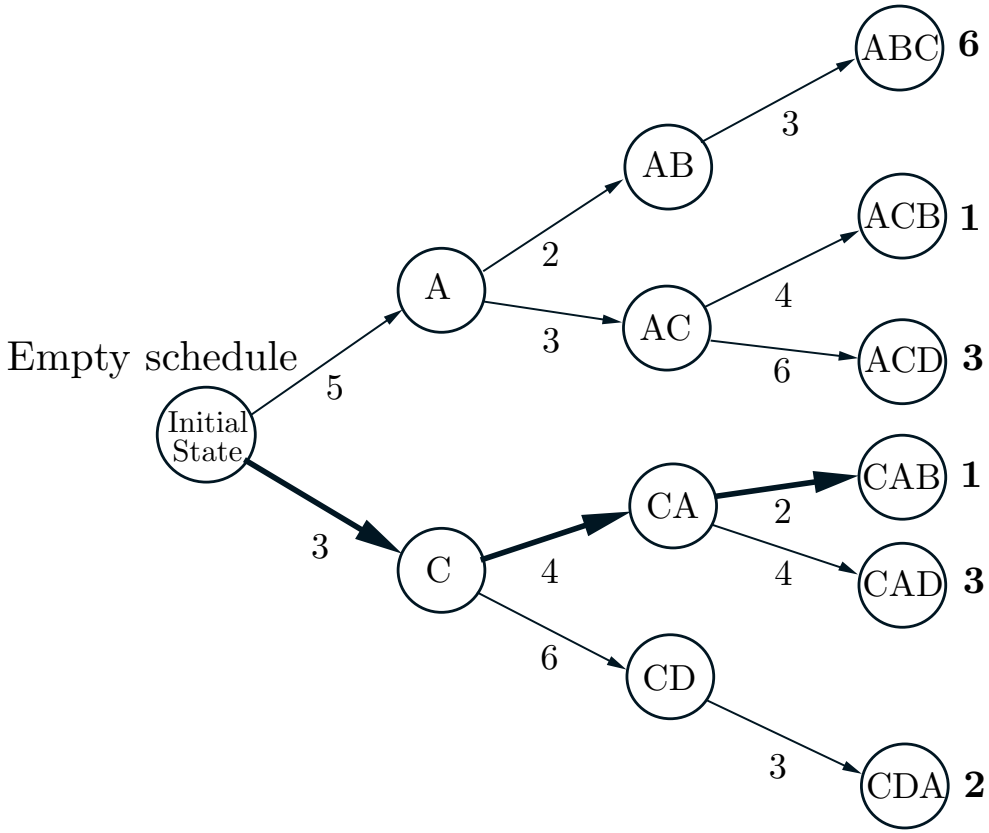
- Linear discrete-time system: $x_{k+1} = Ax_k + Bu_k + w_k, \quad k = 0, \ldots, N-1$
- $x_k \in \Re^n$: The state at time k
- $u_k \in \Re^m$: The control at time k (no constraints in the classical version)
- $w_k \in \Re^n$: The disturbance at time k ($w_0, \ldots, w_{N-1}$ are independent random variables with given distribution)

Quadratic Cost Function

$$E\left\{x_N' Q x_N + \sum_{k=0}^{N-1} \left(x_k' Q x_k + u_k' R u_k\right)\right\}$$

where Q and R are positive definite symmetric matrices

Discrete-State Deterministic Scheduling Example

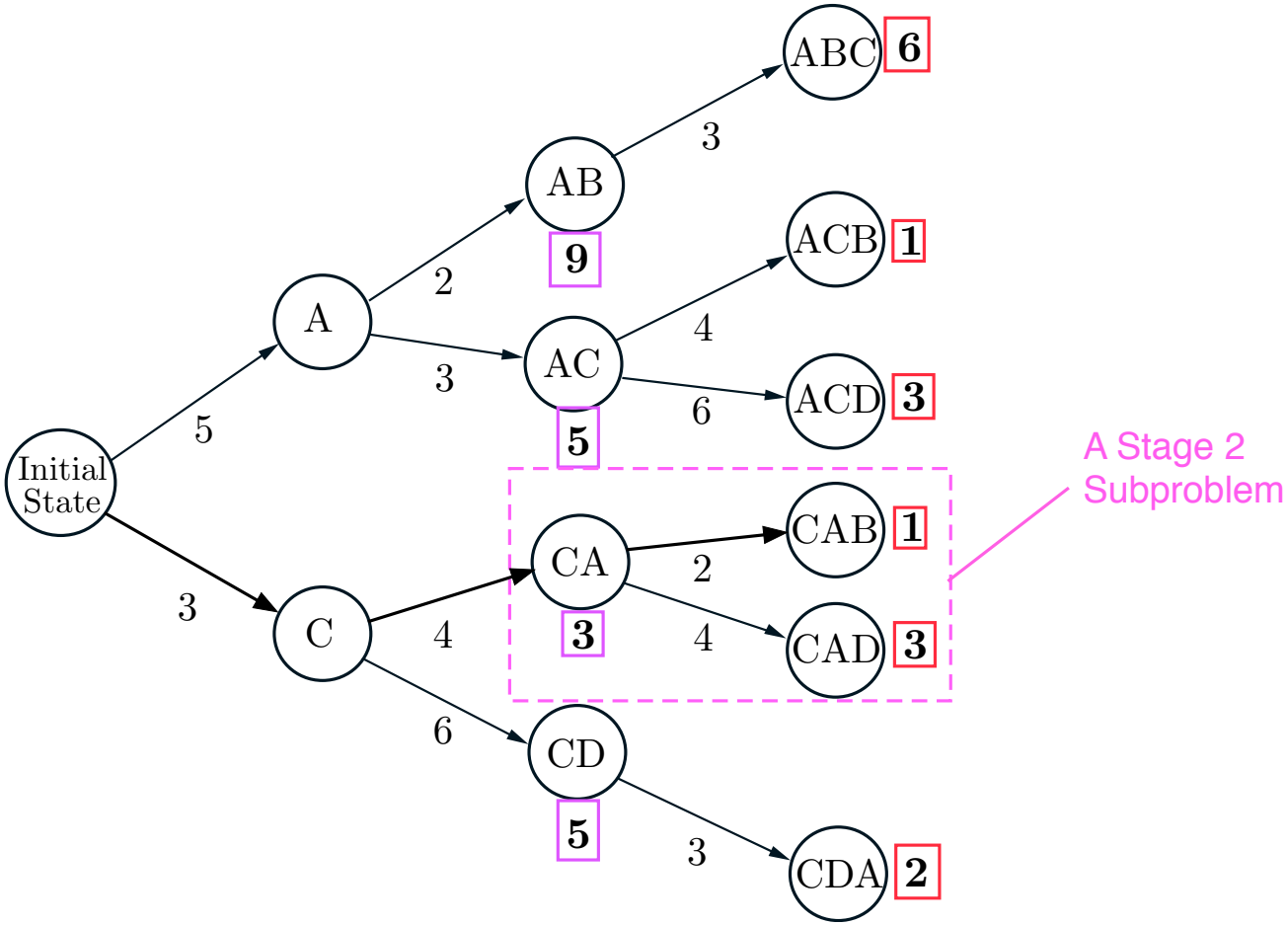


Find optimal sequence of operations A, B, C, D (A must precede B and C must precede D)

DP Problem Formulation

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

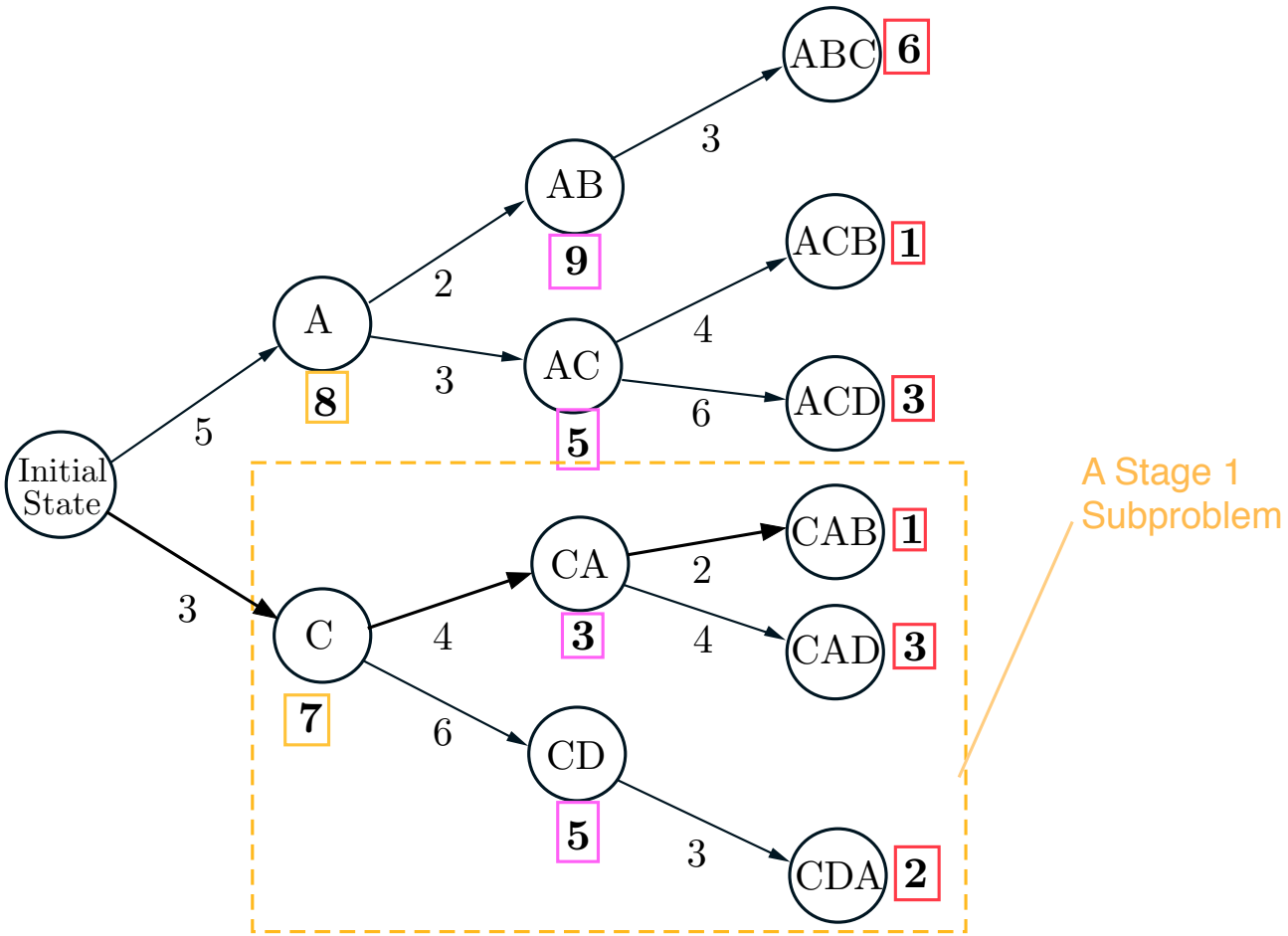
Scheduling Example Algorithm I



Solve the stage 2 subproblems (using the terminal costs)

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

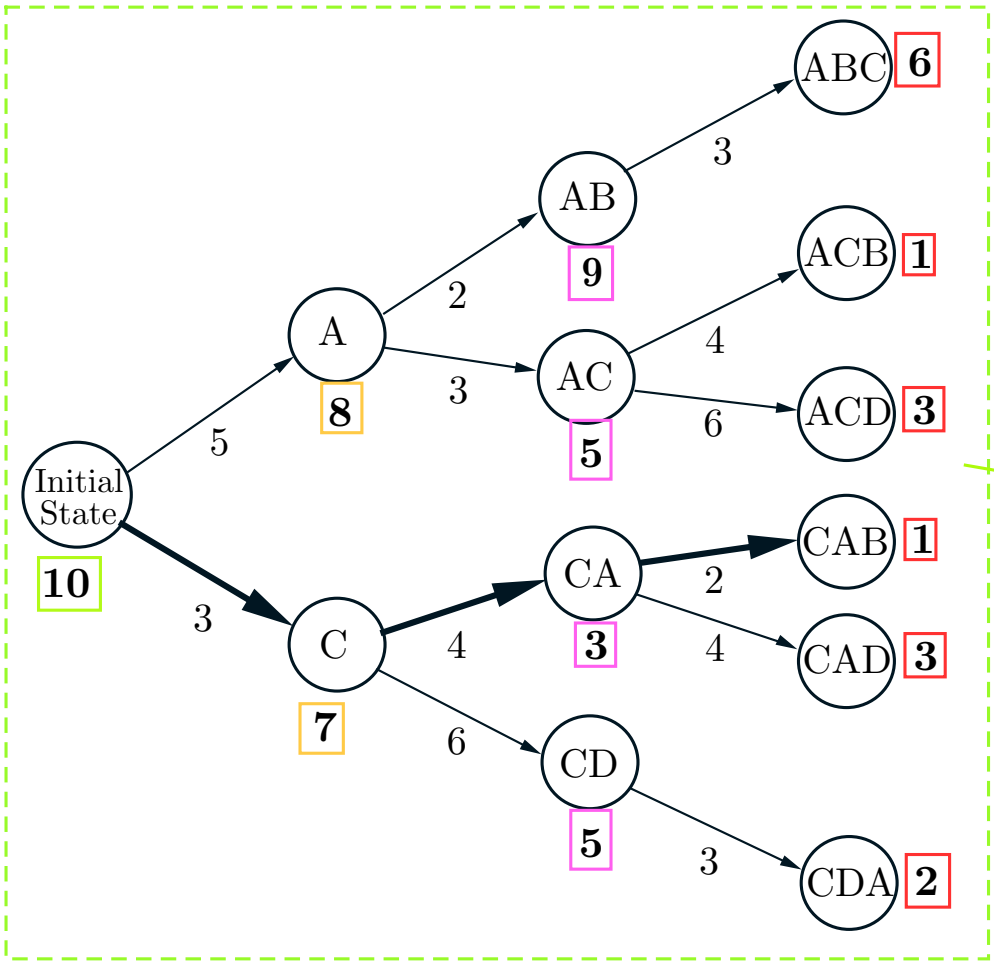
Scheduling Example Algorithm II



Solve the stage 1 subproblems (using the solution of stage 2 subproblems)

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

Scheduling Example Algorithm III



Solve the stage 0 subproblem (using the solution of stage 1 subproblems)

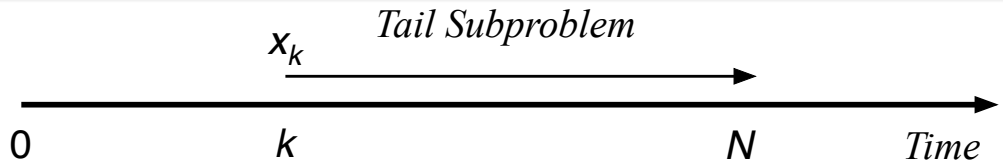
- The stage 0 subproblem is the entire problem
- The optimal value of the stage 0 subproblem is the optimal cost J^*(initial state)
- Construct the optimal sequence going forward

Principle of Optimality

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be an optimal policy
- Consider the "tail subproblem" whereby we are at x_k at time k and wish to minimize the "cost-to-go" from time k to time N

$$E\left\{ g_N(x_N) + \sum_{m=k}^{N-1} g_m\big(x_m, \mu_m(x_m), w_m\big) \right\}$$

Consider the "tail" $\{\mu_k^*, \mu_{k+1}^*, \ldots, \mu_{N-1}^*\}$ of the optimal policy



THE TAIL OF AN OPTIMAL POLICY IS OPTIMAL FOR THE TAIL SUBPROBLEM

DP Algorithm

- Start with the last tail (stage $N-1$) subproblems
- Solve the stage k tail subproblems, using the optimal costs-to-go of the stage $(k+1)$ tail subproblems
- The optimal value of the stage 0 subproblem is the optimal cost J^*(initial state)
- In the process construct the optimal policy

Formal Statement of the DP Algorithm

Computes for all k and states x_k: $J_k(x_k)$: opt. cost of tail problem that starts at x_k

Go backwards, $k = N-1, \ldots, 0$, using

$$J_N(x_N) = g_N(x_N)$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k}\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\}$$

Interpretation: To solve a tail problem that starts at state x_k

Min over u_k the (kth-stage cost + Opt. cost of the tail problem that starts at state x_{k+1})

Notes:

- $J_0(x_0) = J^*(x_0)$: Cost generated at the last step, is equal to the optimal cost
- Let $\mu_k^*(x_k)$ minimize in the right side above for each x_k and k. Then the policy $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is optimal
- Proof by induction

Approximate DP

Approximate DP is primarily motivated by the often ENORMOUS computational demands of exact DP

Some perspectives

- The connection of theory and algorithms (convergence, rate of convergence, complexity, etc) is solid for exact DP and most of optimization
- By contrast, for approximate DP, the connection of theory and algorithms is fragile
- There is a great variety of approximate DP approaches
- Some approximate DP algorithms have been able to solve impressively difficult problems. We often do not fully understand why - a lot of research is ongoing
- There are success stories without theory
- There is theory without success stories
- The theory available is interesting but may involve some assumptions not always satisfied in practice
- Implementation is often an art
- For a given problem: The challenge is how to bring to bear the right mix from a broad array of methods and theoretical ideas