



Introduction on Optimal Control and RL

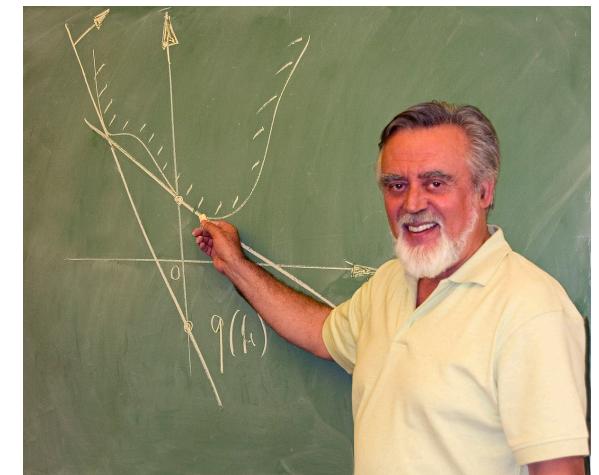
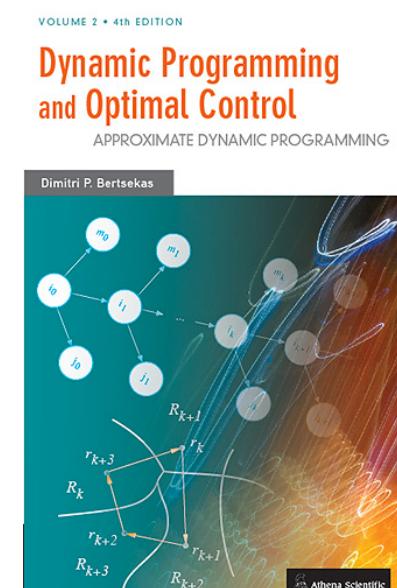
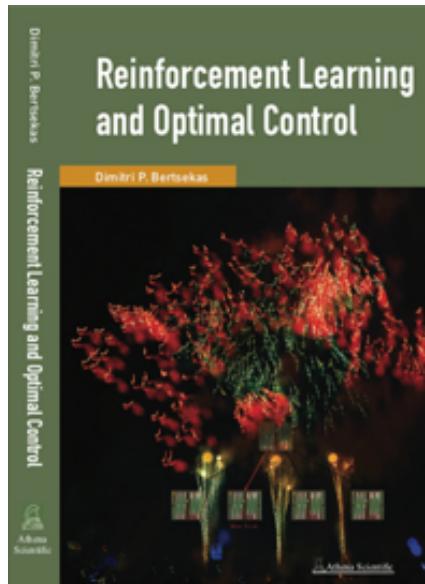
Bolei Zhou

The Chinese University of Hong Kong

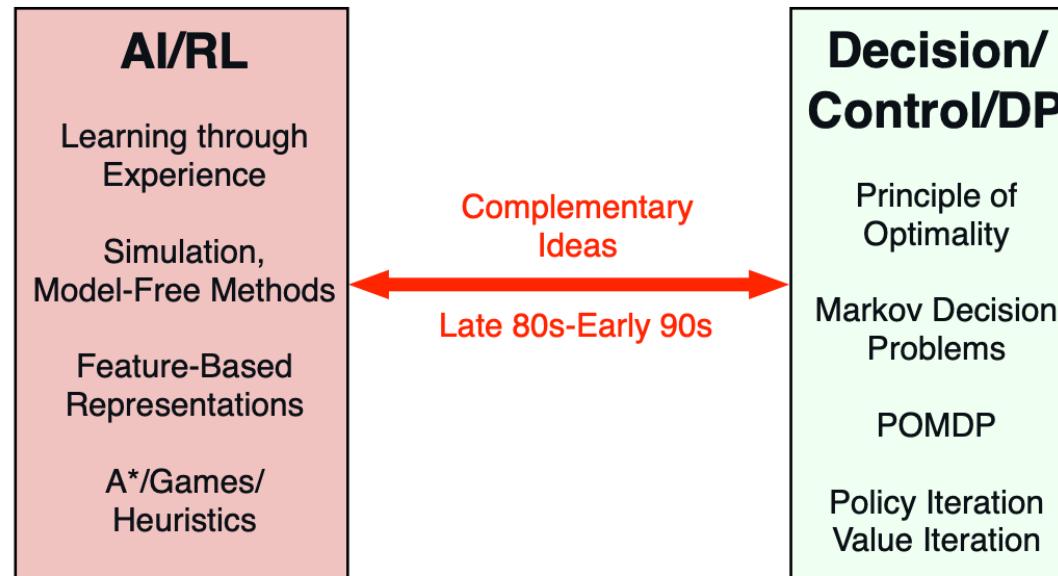
Some slides are from Bertsekas' and his book

Prof. Dimitri P. Bertsekas from MIT

- Researcher on control and optimization
- dynamic programming, optimal control, and reinforcement learning
- <https://web.mit.edu/dimitrib/www/RLbook.html>
- <https://github.com/cuhkrlcourse/bertsekas>



Reinforcement Learning and Optimal Control



Historical highlights

- Exact DP, optimal control (Bellman, Shannon, 1950s ...)
- First major successes: Backgammon programs (Tesauro, 1992, 1996)
- Algorithmic progress, analysis, applications, first books (mid 90s ...)
- Machine Learning, BIG Data, Robotics, Deep Neural Networks (mid 2000s ...)
- AlphaGo and Alphazero (DeepMind, 2016, 2017)

Reinforcement Learning and Optimal Control

RL uses Max/Value, DP uses Min/Cost

- Reward of a stage = (Opposite of) Cost of a stage.
- State value = (Opposite of) State cost.
- Value (or state-value) function = (Opposite of) Cost function.

Controlled system terminology

- Agent = Decision maker or controller.
- Action = Control.
- Environment = Dynamic system.

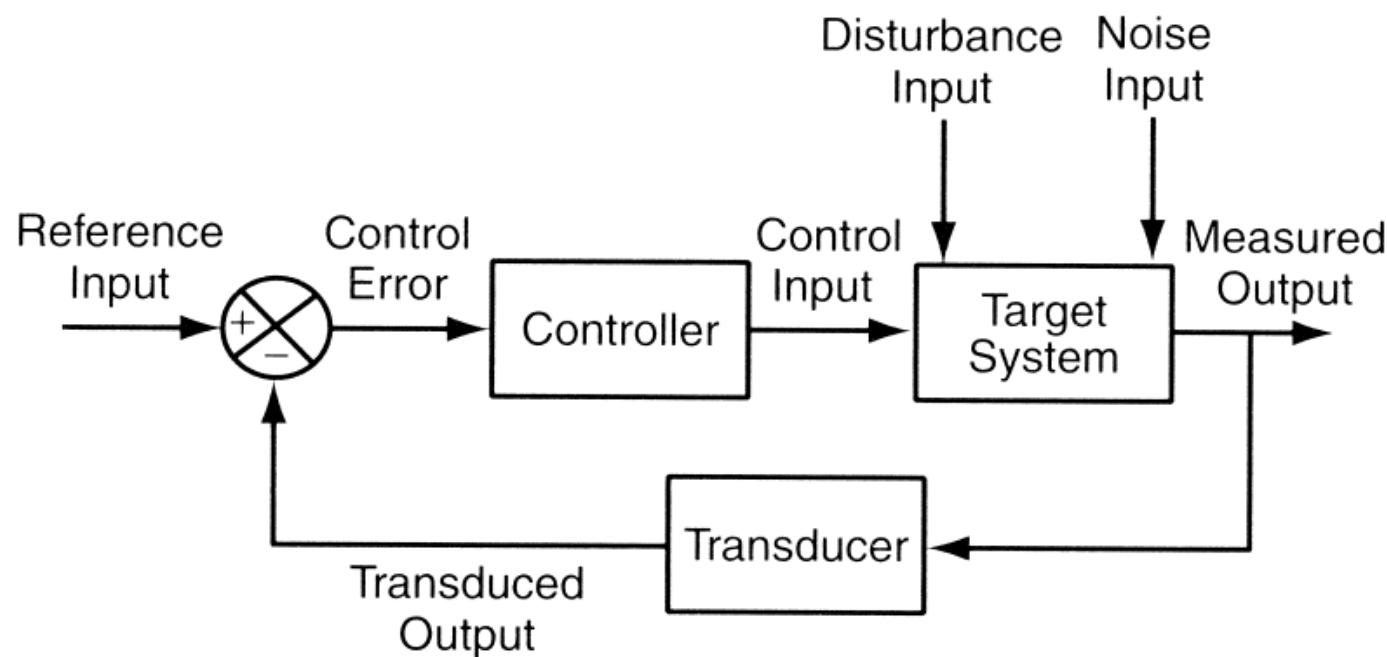
Methods terminology

- Learning = Solving a DP-related problem using simulation.
- Self-learning (or self-play in the context of games) = Solving a DP problem using simulation-based policy iteration.
- Planning vs Learning distinction = Solving a DP problem with model-based vs model-free simulation.

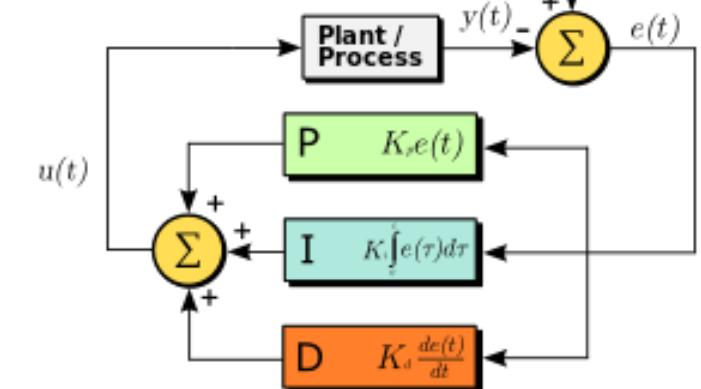
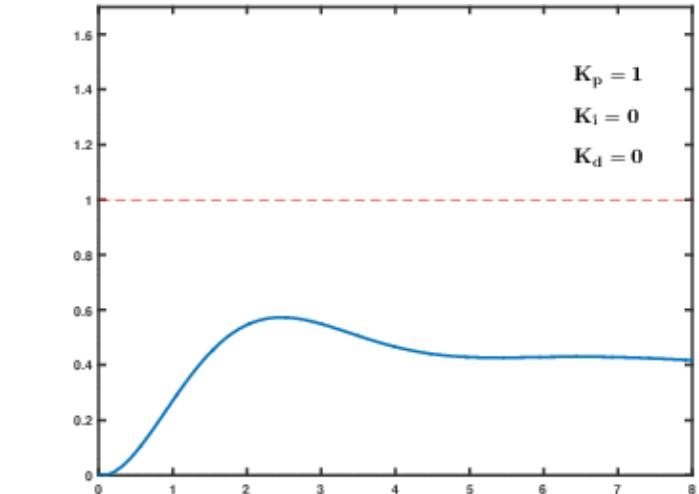
From the Perspective of Control

- Control theory deals with the control of dynamical systems in engineered processes and machines.
- The objective is to develop a control model for controlling such systems using a control action in an optimum manner implementation
- Examples: Keep the temperature, keep the pressure, etc

From the Perspective of Control



Negative feedback control system



$$u(t) = K_P e(t) + K_I \int e(\tau) d\tau + K_D \frac{de(t)}{dt}.$$

Illustration of PID control

Dynamic Programming in Control

- What is a dynamic system

A deterministic DP problem involves a discrete-time dynamic system of the form

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \dots, N-1, \quad (1.1)$$

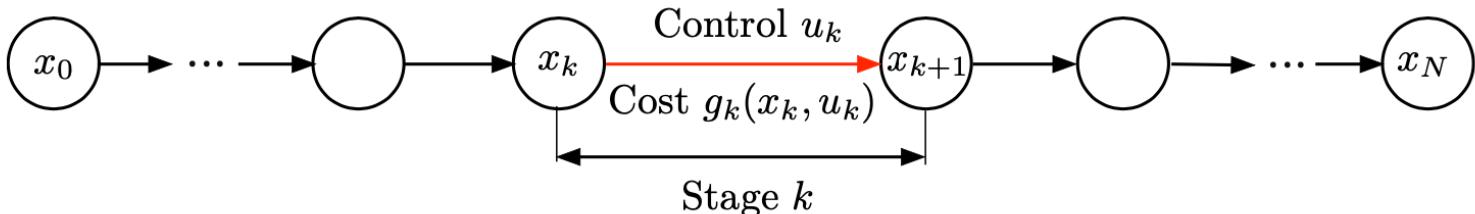
where

k is the time index,

x_k is the state of the system, an element of some space,

u_k is the control or decision variable, to be selected at time k from some given set $U_k(x_k)$ that depends on x_k ,

Deterministic Dynamic Programming



- Discrete-time system:

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

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

- Cost function:

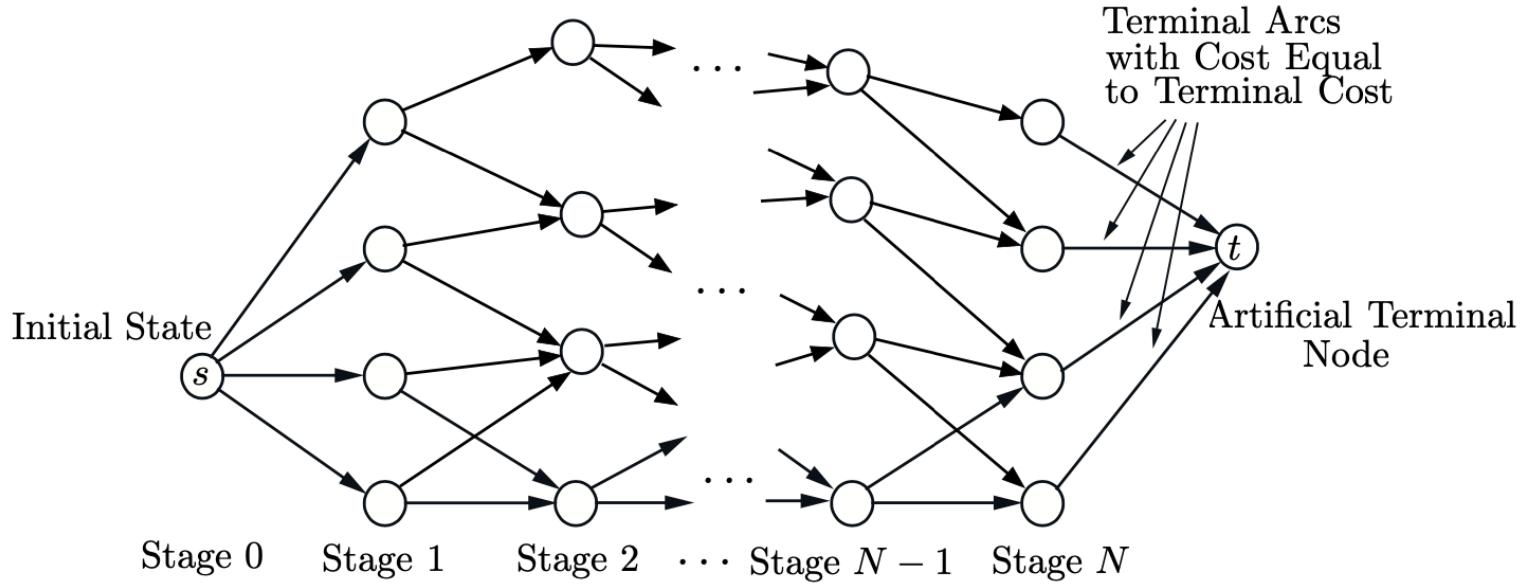
$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

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

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

- Control sequences correspond to paths from start node to end node in the graph
- Optimal cost function $J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0, \dots, N-1}} J(x_0; u_0, \dots, u_{N-1})$

Deterministic Dynamic Programming

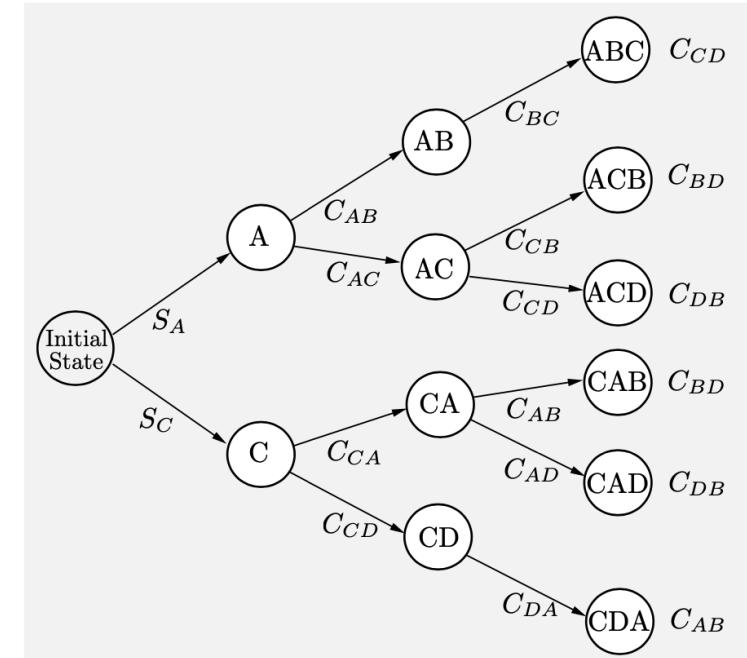


- Nodes correspond to states x_k
- Arcs correspond to state-control pairs (x_k, u_k)
- An arc (x_k, u_k) has start node x_k and end node $x_{k+1} = f_k(x_k, u_k)$
- A control sequence $\{u_0, \dots, u_{N-1}\}$ corresponds to a sequence of arcs from the initial node s to the terminal node t .
- An arc (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 s to t .
- The problem is equivalent to finding a minimum cost/shortest path from s to t .

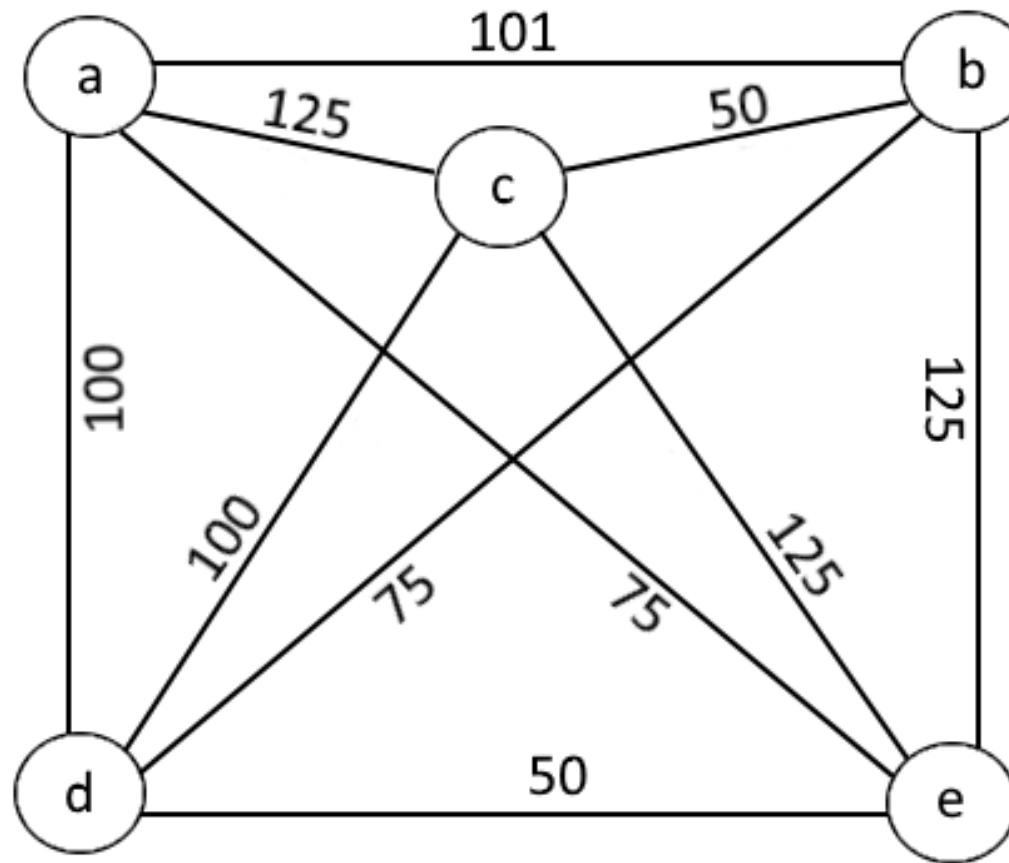
Discrete Optimal Control: a deterministic scheduling problem

- Suppose that to produce a certain product, four operations (A, B, C, D) must be performed on a certain machine.
- We assume that operation B can be performed only after operation A has been performed ($A \rightarrow B$), and operation D can be performed only after operation C has been performed ($C \rightarrow D$)
- A sequence of three decisions
- For example, the cost for sequence ACDB is

$$S_A + C_{AC} + C_{CD} + C_{DB}.$$



Discrete Optimal Control: Travelling salesman problem (TSP)



Continuous-spaces Optimal Control: temperature control

Example 1.1.2 (A Linear-Quadratic Problem)

A certain material is passed through a sequence of N ovens (see Fig. 1.1.4).

Denote

x_0 : initial temperature of the material,

$x_k, k = 1, \dots, N$: temperature of the material at the exit of oven k ,

$u_{k-1}, k = 1, \dots, N$: heat energy applied to the material in oven k .

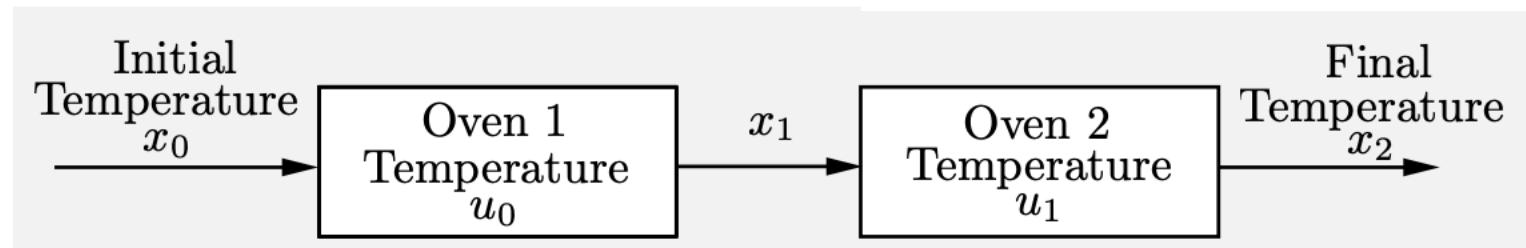
In practice there will be some constraints on u_k , such as nonnegativity. However, for analytical tractability one may also consider the case where u_k is unconstrained, and check later if the solution satisfies some natural restrictions in the problem at hand.

We assume a system equation of the form

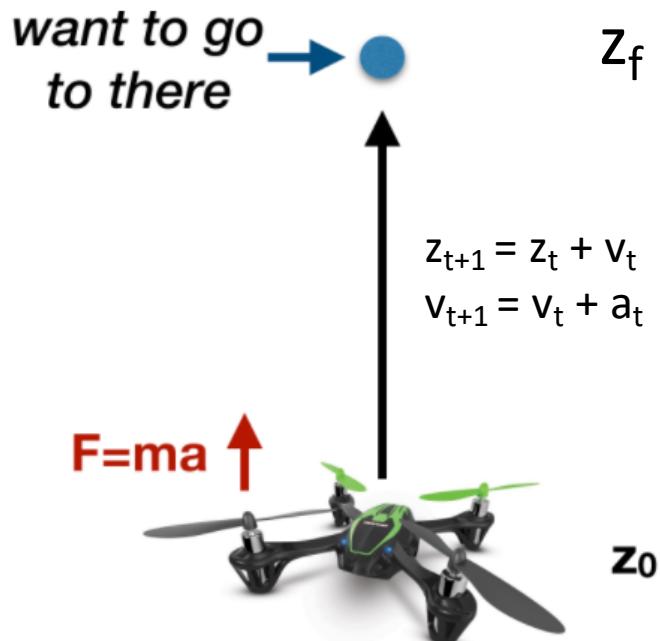
$$x_{k+1} = (1 - a)x_k + au_k, \quad k = 0, 1, \dots, N - 1,$$

where a is a known scalar from the interval $(0, 1)$. The objective is to get the final temperature x_N close to a given target T , while expending relatively little energy. We express this with a cost function of the form

$$r(x_N - T)^2 + \sum_{k=0}^{N-1} u_k^2,$$



Continuous-spaces Optimal Control: quadcopter control



F is force
a is acceleration
v is velocity
z is vertical position

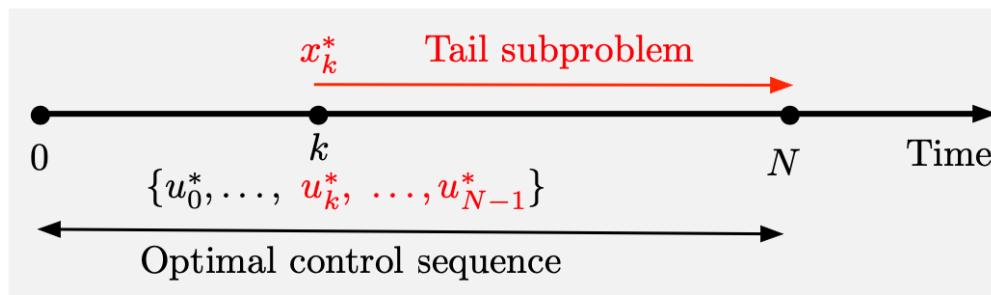
Dynamic Programming: the principle of optimality

Principle of Optimality

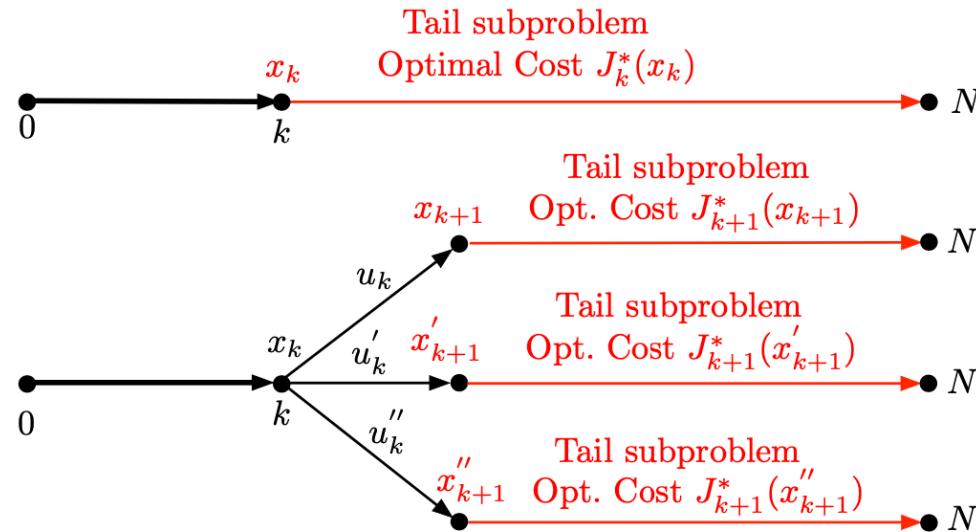
Let $\{u_0^*, \dots, u_{N-1}^*\}$ be an optimal control sequence, which together with x_0 determines the corresponding state sequence $\{x_1^*, \dots, x_N^*\}$ via the system equation (1.1). Consider the subproblem whereby we start at x_k^* at time k and wish to minimize the “cost-to-go” from time k to time N ,

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N),$$

over $\{u_k, \dots, u_{N-1}\}$ with $u_m \in U_m(x_m)$, $m = k, \dots, N - 1$. Then the truncated optimal control sequence $\{u_k^*, \dots, u_{N-1}^*\}$ is optimal for this subproblem.



Dynamic Programming



Consider solution of the **tail subproblem** that starts at x_k at time k and minimizes over $\{u_k, \dots, u_{N-1}\}$ the “cost-to-go” from k to N ,

$$g_k(x_k, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N), \quad \text{Optimal Cost-to-Go: } J_k^*(x_k)$$

To solve it, choose u_k to min (1st stage cost + Optimal tail problem cost) or

$$\min_{u_k \in U_k(x_k)} [g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k))]$$

Dynamic Programming: 2 steps

Going backward

DP Algorithm for Deterministic Finite Horizon Problems

Start with

$$J_N^*(x_N) = g_N(x_N), \quad \text{for all } x_N, \quad (1.3)$$

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

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} [g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k))], \quad \text{for all } x_k. \quad (1.4)$$

Compute optimal cost-to-go function J_k^*

Going forward

Construction of Optimal Control Sequence $\{u_0^*, \dots, u_{N-1}^*\}$

Set

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} [g_0(x_0, u_0) + J_1^*(f_0(x_0, u_0))],$$

and

$$x_1^* = f_0(x_0, u_0^*).$$

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

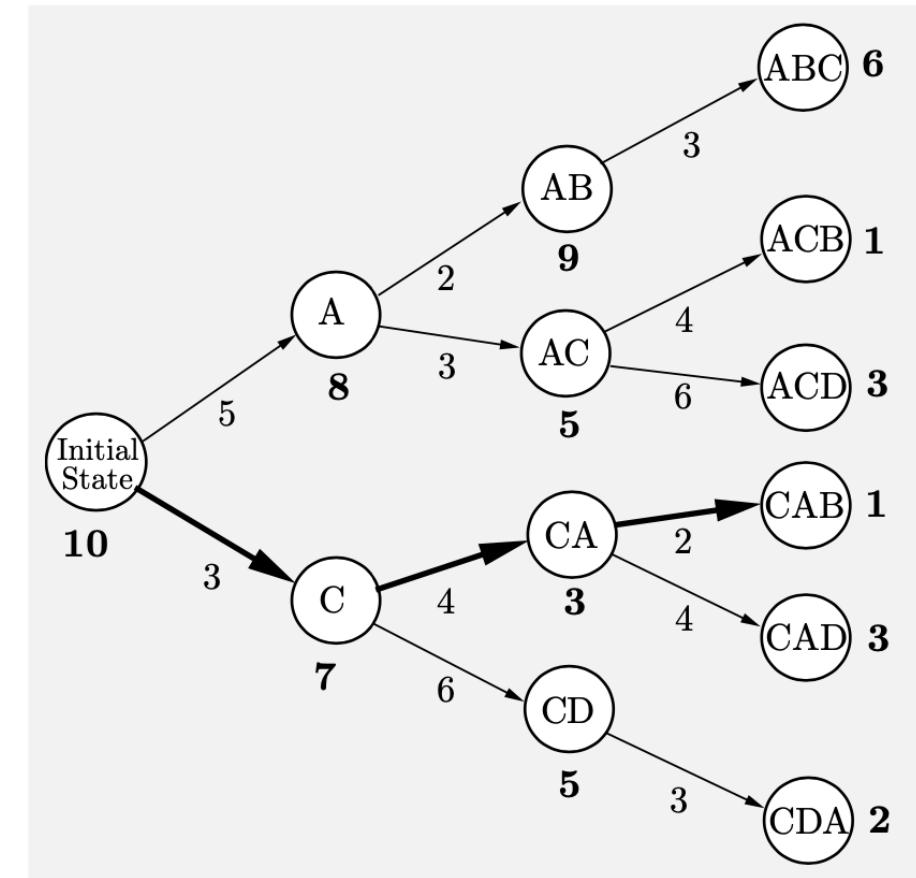
$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} [g_k(x_k^*, u_k) + J_{k+1}^*(f_k(x_k^*, u_k))], \quad (1.7)$$

and

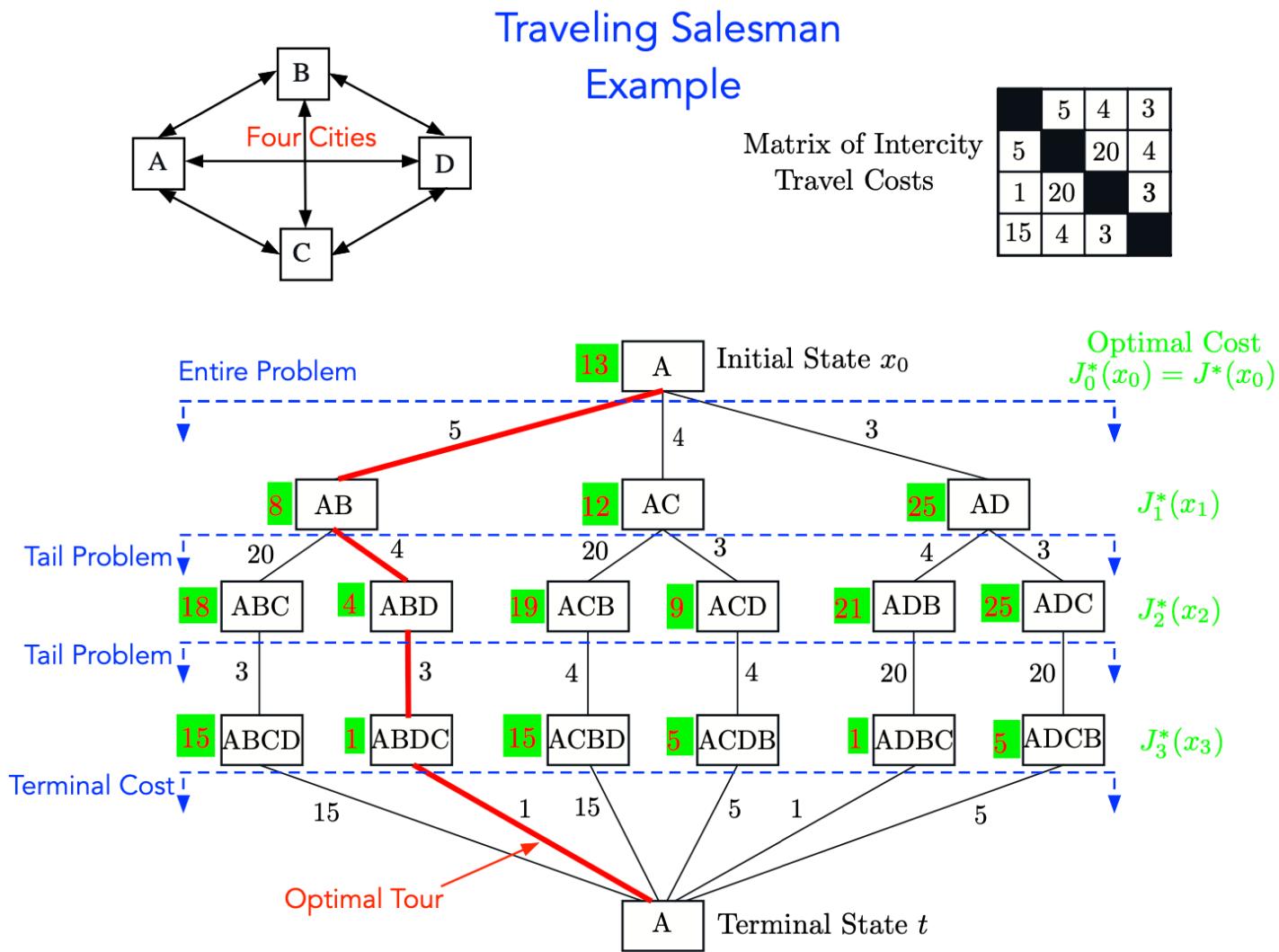
$$x_{k+1}^* = f_k(x_k^*, u_k^*). \quad (1.8)$$

Solving the scheduling problem with DP

- The DP algorithm is based on this idea: it proceeds sequentially, by solving all the tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length.



Solving Traveling Salesman problem with DP



Solving the Linear Quadratic Optimal Control

- Analytical solution for the temperature control

As defined in Example 1.1.2, the terminal cost is

$$g_2(x_2) = r(x_2 - T)^2.$$

Thus the DP algorithm starts with

$$J_2^*(x_2) = g_2(x_2) = r(x_2 - T)^2,$$

For the next-to-last stage, we have [cf. Eq. (1.4)]

$$J_1^*(x_1) = \min_{u_1} [u_1^2 + J_2^*(x_2)] = \min_{u_1} \left[u_1^2 + J_2^*((1-a)x_1 + au_1) \right].$$

Substituting the previous form of J_2^* , we obtain

$$J_1^*(x_1) = \min_{u_1} \left[u_1^2 + r((1-a)x_1 + au_1 - T)^2 \right]. \quad (1.21)$$

This minimization will be done by setting to zero the derivative with respect to u_1 . This yields

$$0 = 2u_1 + 2ra((1-a)x_1 + au_1 - T),$$

and by collecting terms and solving for u_1 , we obtain the optimal temperature for the last oven as a function of x_1 :

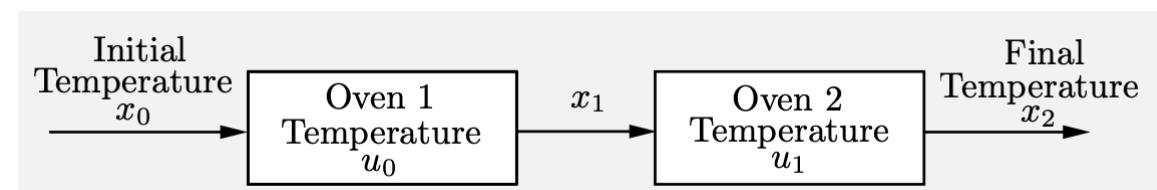
$$\mu_1^*(x_1) = \frac{ra(T - (1-a)x_1)}{1 + ra^2}. \quad (1.22)$$

By substituting the optimal u_1 in the expression (1.21) for J_1^* , we obtain

$$\begin{aligned} J_1^*(x_1) &= \frac{r^2 a^2 ((1-a)x_1 - T)^2}{(1 + ra^2)^2} + r \left((1-a)x_1 + \frac{ra^2(T - (1-a)x_1)}{1 + ra^2} - T \right)^2 \\ &= \frac{r^2 a^2 ((1-a)x_1 - T)^2}{(1 + ra^2)^2} + r \left(\frac{ra^2}{1 + ra^2} - 1 \right)^2 ((1-a)x_1 - T)^2 \\ &= \frac{r((1-a)x_1 - T)^2}{1 + ra^2}. \end{aligned}$$

We now go back one stage. We have [cf. Eq. (1.4)]

$$J_0^*(x_0) = \min_{u_0} [u_0^2 + J_1^*(x_1)] = \min_{u_0} \left[u_0^2 + J_1^*((1-a)x_0 + au_0) \right],$$



Solving the Linear Quadratic Optimal Control

- Analytical solution for the temperature control

and by substituting the expression already obtained for J_1^* , we have

$$J_0^*(x_0) = \min_{u_0} \left[u_0^2 + \frac{r((1-a)^2 x_0 + (1-a)a u_0 - T)^2}{1+ra^2} \right].$$

We minimize with respect to u_0 by setting the corresponding derivative to zero. We obtain

$$0 = 2u_0 + \frac{2r(1-a)a((1-a)^2 x_0 + (1-a)a u_0 - T)}{1+ra^2}.$$

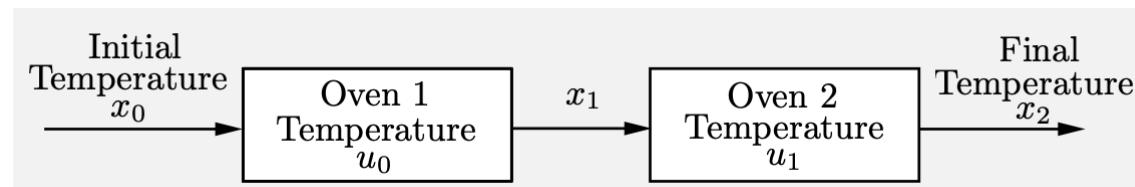
This yields, after some calculation, the optimal temperature of the first oven:

$$\mu_0^*(x_0) = \frac{r(1-a)a(T - (1-a)^2 x_0)}{1+ra^2(1+(1-a)^2)}. \quad (1.23)$$

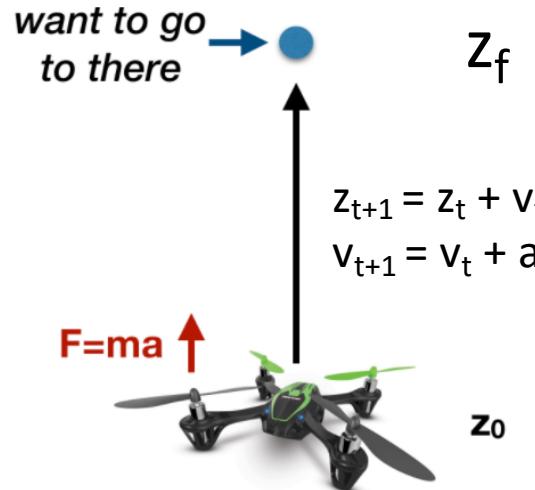
The optimal cost is obtained by substituting this expression in the formula for J_0^* . This leads to a straightforward but lengthy calculation, which in the end yields the rather simple formula

$$J_0^*(x_0) = \frac{r((1-a)^2 x_0 - T)^2}{1+ra^2(1+(1-a)^2)}.$$

what simplifies the solution is the quadratic nature of the cost and the linearity of the system equation.



Quadrotor control problem with analytical solution



F is force
a is acceleration
v is velocity
z is vertical position

control objective. If we want the quadrotor to fly to location z_f , we could specify

$$R_t[x_t, u_t] = |z_t - z_f|$$

or

$$R_t[x_t, u_t] = (z_t - z_f)^2$$

or

$$R_t[x_t, u_t] = \begin{cases} 1 & z_t = z_f \\ 0 & \text{otherwise} \end{cases}$$

Note that the quadrotor dynamics we derived from Newton's Laws are linear. The state of the quadrotor is its vertical position and velocity $x_t = [z_t; v_t]$. The input u_t is the propeller force. Newton's Laws written in matrix form are thus

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} (u_t - g)$$

Thus linear dynamic system: $x_{t+1} = Ax_t + Bu_t$,

Quadrotor control problem with analytical solution

maximize _{u_t} $\mathbb{E}_{e_t} [\sum_{t=0}^N R_t[x_t, u_t]]$
subject to $x_{t+1} = f(x_t, u_t, e_t)$
 (x_0 given).

minimize _{u_t, x_t} $\frac{1}{2} \sum_{t=0}^N \{x_t^T Q x_t + u_t^T R u_t\}$
 $+ \frac{1}{2} x_{N+1}^T S x_{N+1},$

subject to $x_{t+1} = Ax_t + Bu_t,$
 for $t = 0, 1, \dots, N,$
 (x_0 given).

The Lagrangian for the LQR problem has the form:

$$\begin{aligned}\mathcal{L}(x, u, p) := \sum_{t=0}^N & \left[\frac{1}{2} x_t^T Q x_t + \frac{1}{2} u_t^T R u_t \right. \\ & - p_{t+1}^T (x_{t+1} - Ax_t - Bu_t) \Big] \\ & + \frac{1}{2} x_{N+1}^T S x_{N+1}.\end{aligned}$$

After some derivation we can compute u_t as:

$$u_t = -(R + B^T M_{t+1} B)^{-1} B^T M_{t+1} A x_t.$$

DP Algorithm: formal statement

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

Start with

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

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

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

The 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 $\{u_0^*, \dots, u_{N-1}^*\}$

Start with

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

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

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

Problem with exact DP

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

Start with

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

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

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

The optimal cost $J^*(x_0)$ is obtained at the last step: $J_N^*(x_N) = J^*(x_0)$.

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

Start with

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

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

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

Approximation idea: Replace J_k^* with an approximation \tilde{J}_k (possibly a neural net).

Approximation in Value Space

- The construction of suitable approximate cost-to-go functions \tilde{J}_k

Approximation in Value Space - Use of \tilde{J}_k in Place of J_k^*

Start with

$$\tilde{u}_0 \in \arg \min_{u_0 \in U_0(x_0)} \left[g_0(x_0, u_0) + \tilde{J}_1(f_0(x_0, u_0)) \right],$$

and set

$$\tilde{x}_1 = f_0(x_0, \tilde{u}_0).$$

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

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(\tilde{x}_k)} \left[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}(f_k(\tilde{x}_k, u_k)) \right], \quad (1.9)$$

and

$$\tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k). \quad (1.10)$$

Q-Factors and Q-Learning

Approximation in Value Space - Use of \tilde{J}_k in Place of J_k^*

Start with

$$\tilde{u}_0 \in \arg \min_{u_0 \in U_0(x_0)} [g_0(x_0, u_0) + \tilde{J}_1(f_0(x_0, u_0))],$$

and set

$$\tilde{x}_1 = f_0(x_0, \tilde{u}_0).$$

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

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(\tilde{x}_k)} [g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}(f_k(\tilde{x}_k, u_k))], \quad (1.9)$$

and

$$\tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k). \quad (1.10)$$

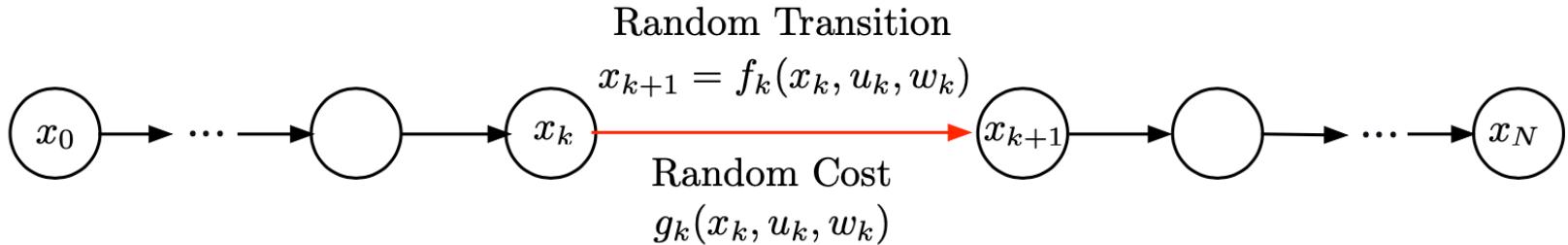
Q-factor of (x_k, u_k) :

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + \tilde{J}_{k+1}(f_k(x_k, u_k)),$$

Optimal Q-factor of (x_k, u_k) :

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k)).$$

Stochastic DP Problems



- Stochasticity in the form of a **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** $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, where μ_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 $\pi = \{\mu_0, \dots, \mu_{N-1}\}$ the cost

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

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

Stochastic DP Algorithm

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

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

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

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

Go forward to implement the optimal policy, given J_1^*, \dots, J_{N-1}^*

Sequentially, starting with x_0 , observe x_k and apply

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

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

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

DP and Approximation in Value Space

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

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

$$J_k(x_k) = \min_{u_k} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

$J_k(x_k)$: Optimal cost-to-go starting from state x_k

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

Approximation in value space: Use an approximate cost-to-go function \tilde{J}_{k+1}

$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

There is also a multistep lookahead version

At state x_k solve an ℓ -step DP problem with terminal cost function approximation $\tilde{J}_{k+\ell}$.
Use the first control in the optimal ℓ -step sequence.

DP and Approximation in Value Space

One-step case at state x_k :

Approximate minimization

$$\min_{u_k} E \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(x_{k+1}) \right\}$$

Approximations:

- Simplify $E\{\cdot\}$
(certainty equivalence)
- Adaptive simulation

“Future”

Computation of \tilde{J}_{k+1} :

- Problem approximation
- Rollout
- Model Predictive Control
- Parametric approximation
- Aggregation

Multistep case at state x_k :

DP minimization

$$\min_{u_k, \mu_{k+1}, \dots, \mu_{k+\ell-1}} E \left\{ g_k(x_k, u_k, w_k) + \sum_{m=k+1}^{k+\ell-1} g_k(x_m, \mu_m(x_m), w_m) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right\}$$

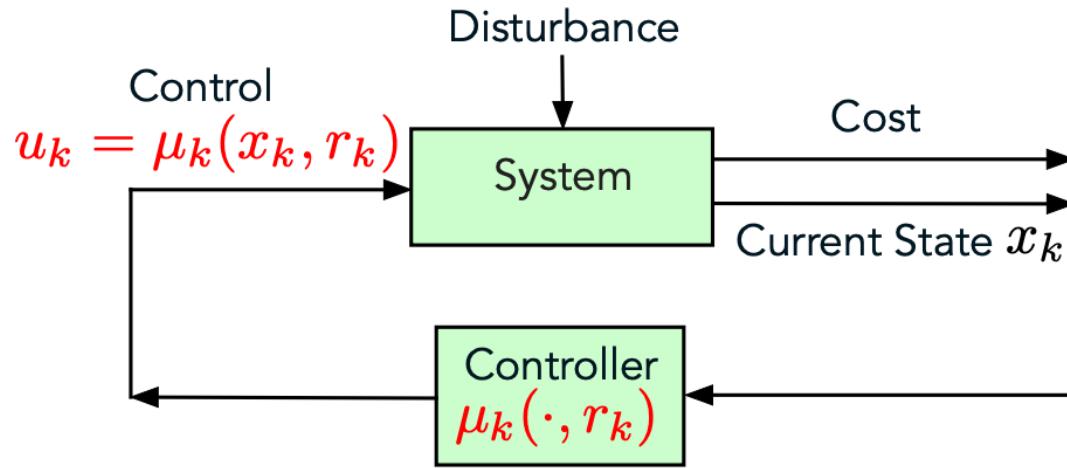
First ℓ Steps

“Future”

Lookahead Minimization

Cost-to-go
Approximation

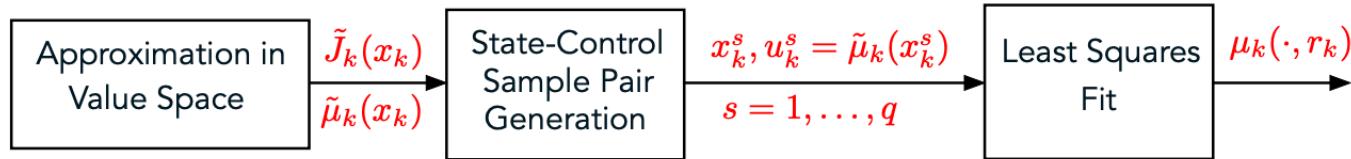
Approximation in Policy Space: The Major Alternative to Approximation in Value Space



- Idea: Select the policy by optimization over a suitably restricted class of policies.
- The restricted class is usually a parametric family of policies $\mu_k(x_k, r_k)$, $k = 0, \dots, N - 1$, of some form, where r_k is a parameter (e.g., a neural net).
- Important advantage once the parameters r_k are computed: The computation of controls during on-line operation of the system is often much easier; i.e.,

At state x_k apply $u_k = \mu_k(x_k, r_k)$

Approximation in Policy Space: The Major Alternative to Approximation in Value Space



- Compute approximate cost-to-go functions \tilde{J}_{k+1} , $k = 0, \dots, N - 1$.
- This defines a suboptimal policy $\tilde{\mu}_k$, $k = 0, \dots, N - 1$, through

$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} E \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

- Generate a training set consisting of a large number q of sample pairs (x_k^s, u_k^s) , $s = 1, \dots, q$, where $u_k^s = \tilde{\mu}_k(x_k^s)$. Approximate $\tilde{\mu}_k$ using some form of parametric least squares fit.
- Example: Introduce a parametric family of policies $\mu_k(x_k, r_k)$, $k = 0, \dots, N - 1$, of some form, where r_k is a parameter vector. Then obtain r_k by

$$r_k \in \arg \min_r \sum_{s=1}^q \|u_k^s - \mu_k(x_k^s, r)\|^2$$

- Some other possibilities: Nearest neighbor optimization (use the control u_k^s of the sampled state x_k^s nearest to x_k), or interpolation.

Model-Based Versus Model-Free Implementation for Stochastic Problems

Our use of the term “**model-free**”: A method is called model-free if it involves calculations of expected values using **Monte Carlo simulation**.

Principal example

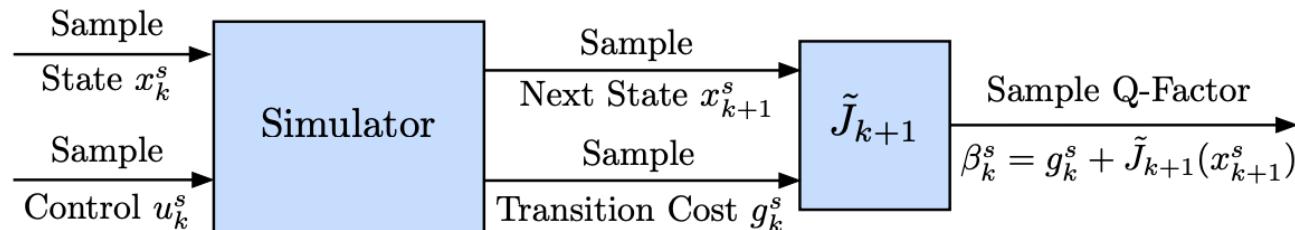
- Calculate \tilde{J}_k in **model-free fashion** (rollout is a possibility to be discussed later).
- Implement **model-free control minimization** with one of three possible methods:
 - ▶ (1) On-line, at state x_k calculate by simulation the needed Q-factors for lookahead minimization

$$\tilde{Q}_k(x_k, u_k) = E \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

- and choose u_k that minimizes $\tilde{Q}_k(x_k, u_k)$
- ▶ (2) Off-line “train” a parametric class of policies $\tilde{\mu}_k(x_k, r_k)$ by approximation in policy space on top of approximation in value space (see previous slide)
 - ▶ (3) Off-line “train” a parametric family of Q-factors $\tilde{Q}_k(x_k, u_k, r_k)$ and choose u_k that minimizes $\tilde{Q}_k(x_k, u_k, r_k)$ (see next slide)

Model-Free Q-Factor Parametric Approximation

- Heavily involve Monte-Carlo simulation



- Use the simulator to collect a large number of “representative” samples of state-control-successor states-stage cost quadruplets $(x_k^s, u_k^s, x_{k+1}^s, g_k^s)$, and corresponding sample Q-factors

$$\beta_k^s = g_k^s + \tilde{J}_{k+1}(x_{k+1}^s), \quad s = 1, \dots, q$$

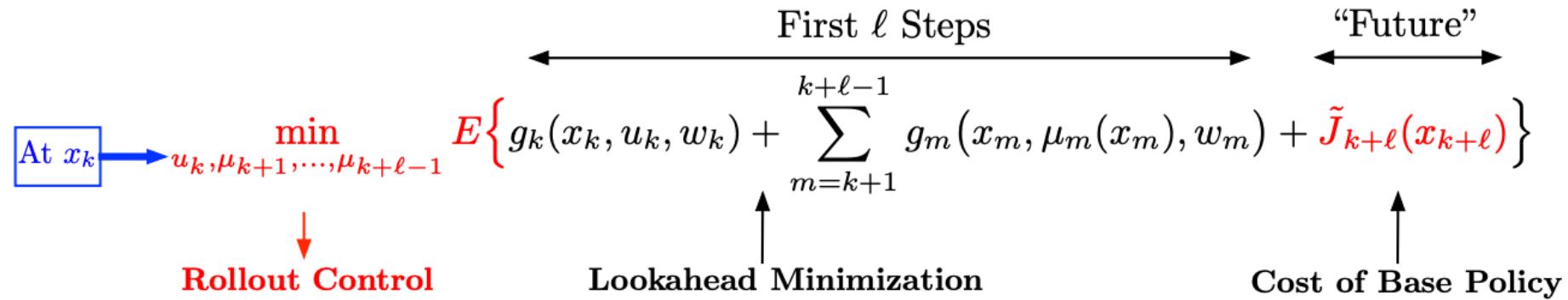
- Introduce a parametric family of Q-factors $\tilde{Q}_k(x_k, u_k, r_k)$.
- Determine the parameter vector \bar{r}_k by the least-squares fit

$$\bar{r}_k \in \arg \min_{r_k} \sum_{s=1}^q (\tilde{Q}_k(x_k^s, u_k^s, r_k) - \beta_k^s)^2$$

- Use the policy

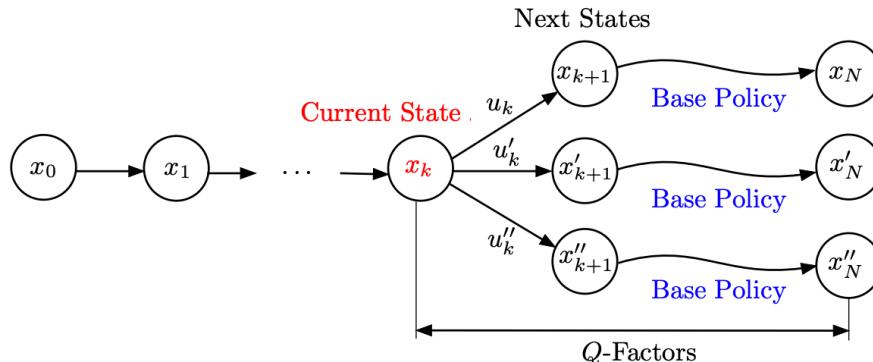
$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k, \bar{r}_k)$$

Rollout: start with a base policy, get a better policy



Use the cost of the base/suboptimal policy at the end of ℓ -step lookahead

Deterministic Rollout ($\ell = 1$) - Avoids Expensive Simulation



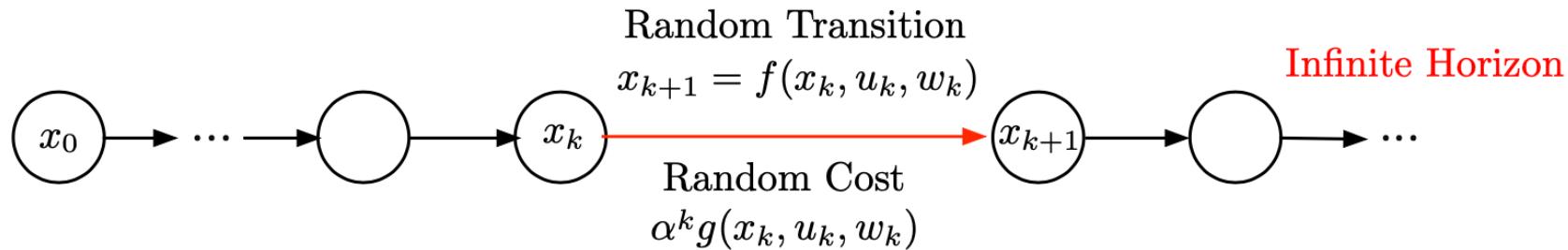
- At state x_k , for every pair (x_k, u_k) , $u_k \in U_k(x_k)$, we generate a Q-factor

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + H_{k+1}(f_k(x_k, u_k))$$

using the base policy $[H_{k+1}(x_{k+1})]$ is the base policy cost starting from x_{k+1} .

- We select the control u_k with minimal Q-factor.
- We move to the next state x_{k+1} , and continue.

Stochastic DP Algorithm: Infinite Horizon Extension



Infinite number of stages, and stationary system and cost

- System $x_{k+1} = f(x_k, u_k, w_k)$ with state, control, and random disturbance.
- Policies $\pi = \{\mu_0, \mu_1, \dots\}$ with $\mu_k(x) \in U(x)$ for all x and k .
- Optimal cost function $J^*(x_0) = \min_{\pi} J_{\pi}(x_0)$ satisfies Bellman's equation

$$J^*(x) = \min_{u \in U(x)} E\{g(x, u, w) + \alpha J^*(f(x, u, w))\}$$

- **Optimal policy:** Applies at x the minimizing u above, regardless of stage k .
- When there are finitely many states, $i = 1, \dots, n$, Bellman's equation is written in terms of the $i \rightarrow j$ transition probabilities $p_{ij}(u)$ as

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u)(g(i, u, j) + \alpha J^*(j))$$

- **Approximation possibility:** Use \tilde{J} in place of J^* , and approximate $E\{\cdot\}$ and \min_u

Value Iteration

$$J_{k+1}(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + J_k(f(x, u, w)) \right\}, \quad k = 0, 1, \dots, \quad (4.1)$$

The optimal infinite horizon cost is the limit of the corresponding N -stage optimal costs as $N \rightarrow \infty$; i.e.,

$$J^*(x) = \lim_{N \rightarrow \infty} J_N(x) \quad (4.2)$$

for all states x .

The following equation should hold for all states x ,

$$J^*(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + J^*(f(x, u, w)) \right\}. \quad (4.3)$$

Value Iteration for Discounted Problems

Bellman Equation and Value Iteration for Discounted Problems:

For all $i = 1, \dots, n$, we have

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u)(g(i, u, j) + \alpha J^*(j)).$$

For all $i = 1, \dots, n$, and any initial conditions $J_0(1), \dots, J_0(n)$, the VI algorithm generates the sequence $\{J_k\}$ according to

$$J_{k+1}(i) = \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u)(g(i, u, j) + \alpha J_k(j)).$$

Proposition 4.3.1: (Convergence of VI) Given any initial conditions $J_0(1), \dots, J_0(n)$, the sequence $\{J_k(i)\}$ generated by the VI algorithm

$$J_{k+1}(i) = \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u)(g(i, u, j) + \alpha J_k(j)), \quad i = 1, \dots, n,$$

converges to the optimal cost $J^*(i)$ for each i .

DP (Bellman) operators that map the vector J into another vectors

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u) (g(i, u, j) + \alpha J(j)), \quad i = 1, \dots, n, \quad (4.13)$$

and

$$(T_\mu J)(i) = \sum_{j=1}^n p_{ij}(\mu(i)) (g(i, \mu(i), j) + \alpha J(j)), \quad i = 1, \dots, n, \quad (4.14)$$

Fixed point property of the DP mapping:

$$\|J_k - J^*\| \leq \rho^k \|J_0 - J^*\|.$$

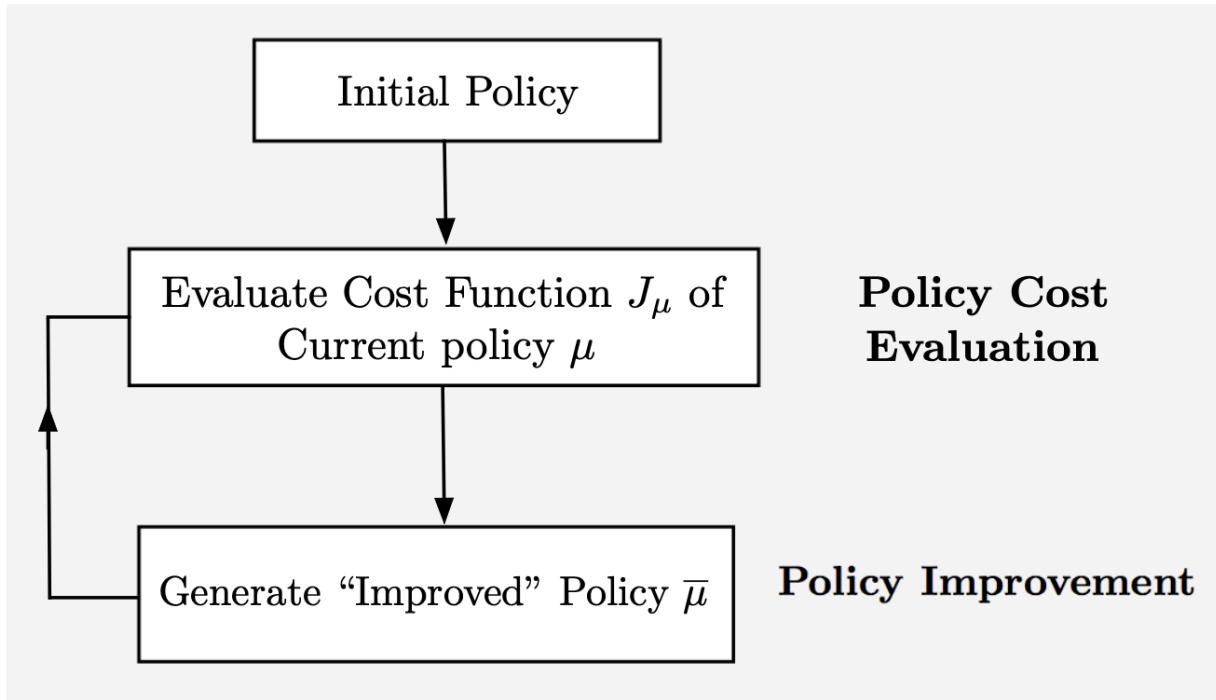
Proposition 4.3.5: (Contraction Property of the DP Operator) The DP operators T and T_μ of Eqs. (4.13) and (4.14) are contraction mappings of modulus α with respect to the maximum norm

$$\|J\| = \max_{i=1, \dots, n} |J(i)|. \quad (4.15)$$

In particular, for any two n -dimensional vectors J and J' , we have

$$\|TJ - TJ'\| \leq \alpha \|J - J'\|, \quad \|T_\mu J - T_\mu J'\| \leq \alpha \|J - J'\|.$$

Policy Iteration



Exact Policy Iteration: Discounted Problems

Given the typical policy μ^k :

Policy evaluation computes $J_{\mu^k}(i)$, $i = 1, \dots, n$, as the solution of the (linear) system of Bellman equations

$$J_{\mu^k}(i) = \sum_{j=1}^n p_{ij}(\mu^k(i)) (g(i, \mu^k(i), j) + \alpha J_{\mu^k}(j)), \quad i = 1, \dots, n, \quad (4.25)$$

(cf. Prop. 4.2.3).

Policy improvement then computes a new policy μ^{k+1} as

$$\mu^{k+1}(i) \in \arg \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u) (g(i, u, j) + \alpha J_{\mu^k}(j)), \quad i = 1, \dots, n. \quad (4.26)$$

The process is repeated with μ^{k+1} used in place of μ^k , unless we have $J_{\mu^{k+1}}(i) = J_{\mu^k}(i)$ for all i , in which case the algorithm terminates with the policy μ^k .

Convergence of exact PI:

$$J_{\mu^{k+1}}(i) \leq J_{\mu^k}(i), \quad \text{for all } i \text{ and } k,$$

Resources

- Chapter 1,2,4 of the Bertsekas' textbook (more theory driven)
 - <https://github.com/cuhkrlcourse/bertsekas>
- Reinforcement learning theory monograph:
 - By Alekh Agarwal, Nan Jiang, Sham Kakade:
https://rltheorybook.github.io/rl_monograph_AJK.pdf
- Ben Recht (Berkeley Professor in control theory): An outsider's tour of reinforcement learning:
<http://www.argmin.net/2018/06/25/outsider-rl/>