

LECTURE SLIDES - DYNAMIC PROGRAMMING

BASED ON LECTURES GIVEN AT THE

MASSACHUSETTS INST. OF TECHNOLOGY

CAMBRIDGE, MASS

FALL 2012

DIMITRI P. BERTSEKAS

These lecture slides are based on the two-volume book: “Dynamic Programming and Optimal Control” Athena Scientific, by D. P. Bertsekas (Vol. I, 3rd Edition, 2005; Vol. II, 4th Edition, 2012); see

<http://www.athenasc.com/dpbook.html>

Two related reference books:

- (1) “Neuro-Dynamic Programming,” Athena Scientific, by D. P. Bertsekas and J. N. Tsitsiklis, 1996
- (2) “Introduction to Probability” (2nd edition), Athena Scientific, by D. P. Bertsekas and J. N. Tsitsiklis, 2008

6.231: DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

- Problem Formulation
- Examples
- The Basic Problem
- Significance of Feedback

DP AS AN OPTIMIZATION METHODOLOGY

- Generic optimization problem:

$$\min_{u \in U} g(u)$$

where u is the optimization/decision variable, $g(u)$ is the cost function, and U is the constraint set

- Categories of problems:
 - **Discrete** (U is finite) or **continuous**
 - **Linear** (g is linear and U is polyhedral) or **nonlinear**
 - **Stochastic or deterministic**: In stochastic problems the cost involves a stochastic parameter w , which is averaged, i.e., it has the form

$$g(u) = E_w \{ G(u, w) \}$$

where w is a random parameter.

- DP can deal with complex stochastic problems where information about w becomes available in stages, and the decisions are also made in stages and make use of this information.

BASIC STRUCTURE OF STOCHASTIC DP

- Discrete-time system

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

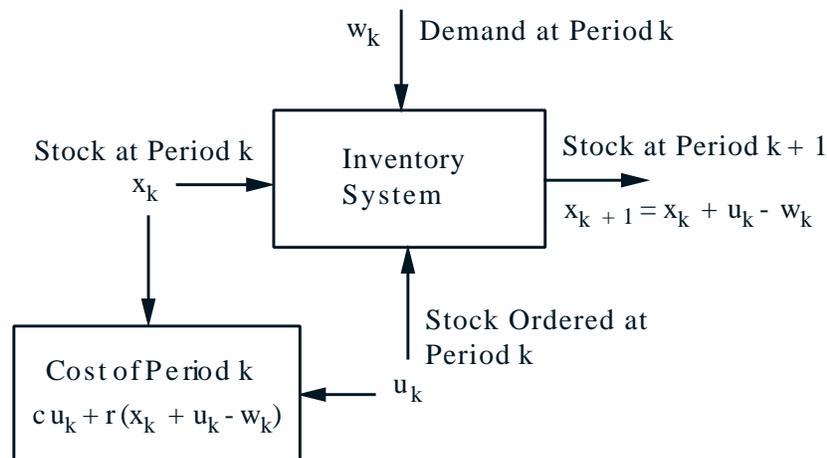
- k : **Discrete time**
 - x_k : **State**; summarizes past information that is relevant for future optimization
 - u_k : **Control**; decision to be selected at time k from a given set
 - w_k : **Random parameter** (also called disturbance or noise depending on the context)
 - N : **Horizon** or number of times control is applied
- Cost function that is additive over time

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

- **Alternative system description:** $P(x_{k+1} \mid x_k, u_k)$

$$x_{k+1} = w_k \quad \text{with} \quad P(w_k \mid x_k, u_k) = P(x_{k+1} \mid x_k, u_k)$$

INVENTORY CONTROL EXAMPLE



- Discrete-time system

$$x_{k+1} = f_k(x_k, u_k, w_k) = x_k + u_k - w_k$$

- Cost function that is additive over time

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

$$= E \left\{ \sum_{k=0}^{N-1} (c u_k + r(x_k + u_k - w_k)) \right\}$$

- Optimization over policies: Rules/functions $u_k = \mu_k(x_k)$ that map states to controls

ADDITIONAL ASSUMPTIONS

- The set of values that the control u_k can take depend at most on x_k and not on prior x or u
- Probability distribution of w_k does not depend on past values w_{k-1}, \dots, w_0 , but may depend on x_k and u_k
 - Otherwise past values of w or x would be useful for future optimization
- Sequence of events envisioned in period k :
 - x_k occurs according to

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$$

- u_k is selected with knowledge of x_k , i.e.,

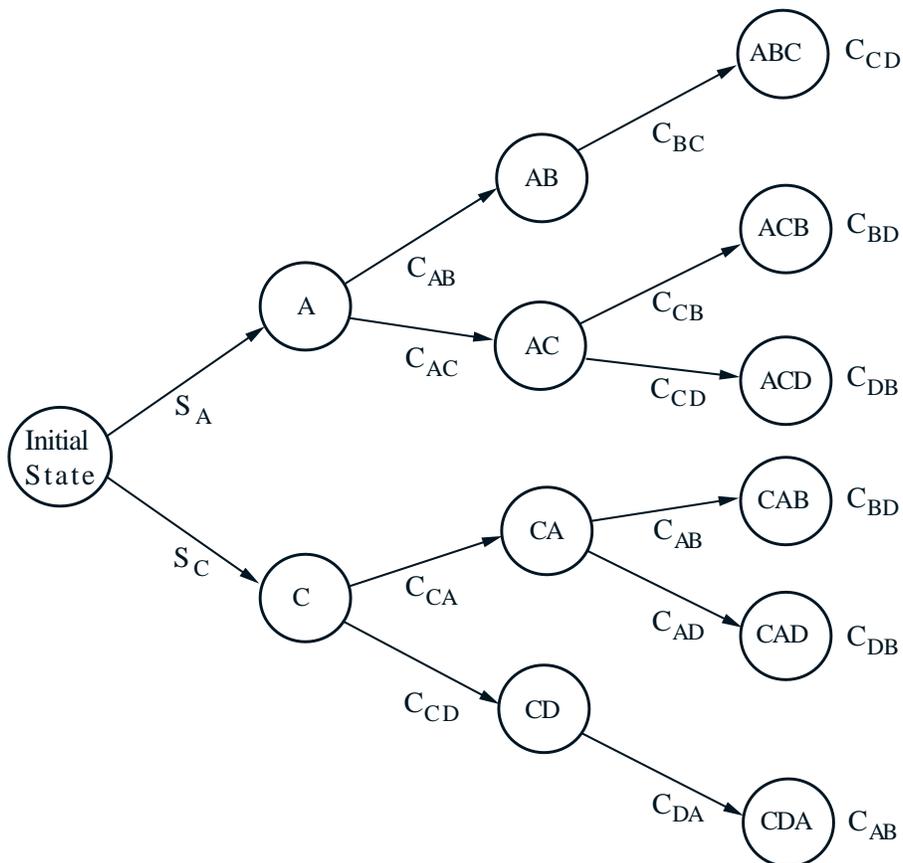
$$u_k \in U_k(x_k)$$

- w_k is random and generated according to a distribution

$$P_{w_k}(x_k, u_k)$$

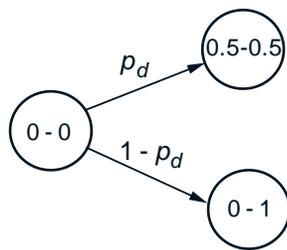
DETERMINISTIC FINITE-STATE PROBLEMS

- Scheduling example: Find optimal sequence of operations A, B, C, D
- A must precede B, and C must precede D
- Given startup cost S_A and S_C , and setup transition cost C_{mn} from operation m to operation n

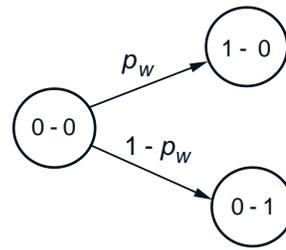


STOCHASTIC FINITE-STATE PROBLEMS

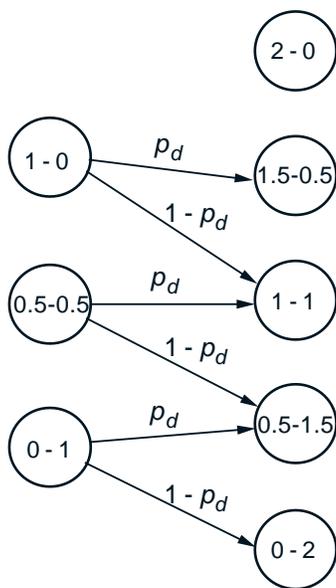
- Example: Find two-game chess match strategy
- *Timid* play draws with prob. $p_d > 0$ and loses with prob. $1 - p_d$. *Bold* play wins with prob. $p_w < 1/2$ and loses with prob. $1 - p_w$



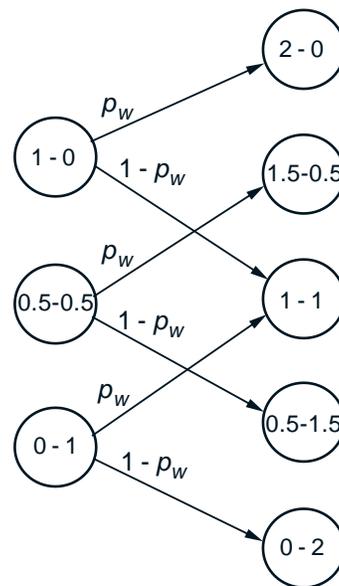
1st Game / Timid Play



1st Game / Bold Play



2nd Game / Timid Play



2nd Game / Bold Play

BASIC PROBLEM

- **System** $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \dots, N-1$
- **Control constraints** $u_k \in U_k(x_k)$
- **Probability distribution** $P_k(\cdot | x_k, u_k)$ of w_k
- **Policies** $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k
- **Expected cost** of π starting at x_0 is

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

- **Optimal cost function**

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

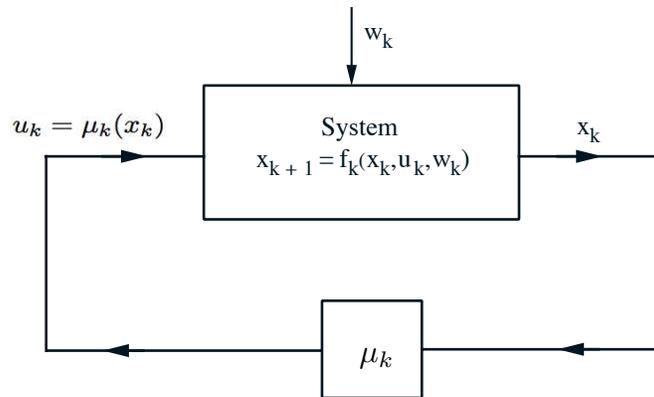
- Optimal policy π^* satisfies

$$J_{\pi^*}(x_0) = J^*(x_0)$$

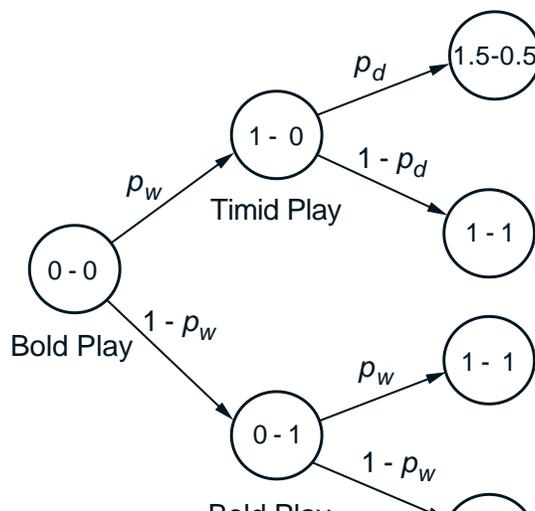
When produced by DP, π^* is independent of x_0 .

SIGNIFICANCE OF FEEDBACK

- Open-loop versus closed-loop policies



- In deterministic problems open loop is as good as closed loop
- Value of information; chess match example
- Example of open-loop policy: Play always bold
- Consider the closed-loop policy: Play timid if and only if you are ahead



VARIANTS OF DP PROBLEMS

- Continuous-time problems
- Imperfect state information problems
- Infinite horizon problems
- Suboptimal control

LECTURE BREAKDOWN

- **Finite Horizon Problems** (Vol. 1, Ch. 1-6)
 - Ch. 1: The DP algorithm (2 lectures)
 - Ch. 2: Deterministic finite-state problems (1 lecture)
 - Ch. 3: Deterministic continuous-time problems (1 lecture)
 - Ch. 4: Stochastic DP problems (2 lectures)
 - Ch. 5: Imperfect state information problems (2 lectures)
 - Ch. 6: Suboptimal control (2 lectures)
- **Infinite Horizon Problems - Simple** (Vol. 1, Ch. 7, 3 lectures)
- **Infinite Horizon Problems - Advanced** (Vol. 2)
 - Chs. 1, 2: Discounted problems - Computational methods (3 lectures)
 - Ch. 3: Stochastic shortest path problems (1 lecture)
 - Chs. 6, 7: Approximate DP (6 lectures)

A NOTE ON THESE SLIDES

- These slides are a teaching aid, not a text
- Don't expect a rigorous mathematical development or precise mathematical statements
- Figures are meant to convey and enhance ideas, not to express them precisely
- Omitted proofs and a much fuller discussion can be found in the texts, which these slides follow

6.231 DYNAMIC PROGRAMMING

LECTURE 2

LECTURE OUTLINE

- The basic problem
- Principle of optimality
- DP example: Deterministic problem
- DP example: Stochastic problem
- The general DP algorithm
- State augmentation

BASIC PROBLEM

- **System** $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \dots, N-1$
- **Control constraints** $u_k \in U_k(x_k)$
- **Probability distribution** $P_k(\cdot | x_k, u_k)$ of w_k
- **Policies** $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k
- **Expected cost** of π starting at x_0 is

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

- **Optimal cost function**

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

- Optimal policy π^* is one that satisfies

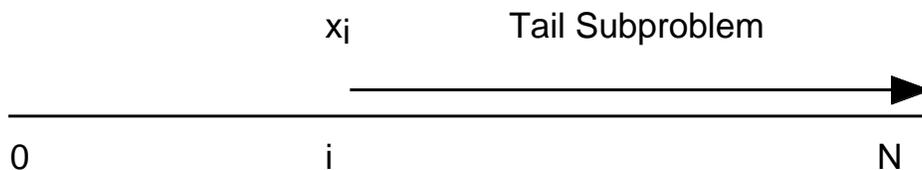
$$J_{\pi^*}(x_0) = J^*(x_0)$$

PRINCIPLE OF OPTIMALITY

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$ be optimal policy
- Consider the “tail subproblem” whereby we are at x_i at time i and wish to minimize the “cost-to-go” from time i to time N

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

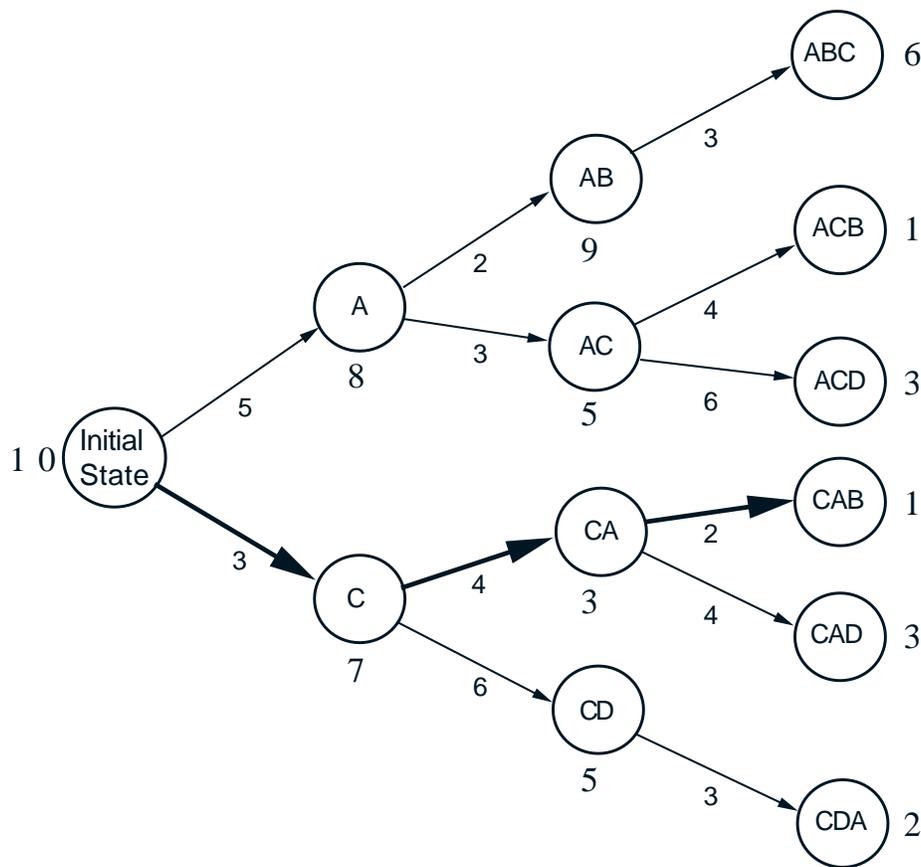
and the “tail policy” $\{\mu_i^*, \mu_{i+1}^*, \dots, \mu_{N-1}^*\}$



- *Principle of optimality*: The tail policy is optimal for the tail subproblem (optimization of the future does not depend on what we did in the past)
- DP first solves ALL tail subproblems of final stage
- At the generic step, it solves ALL tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length

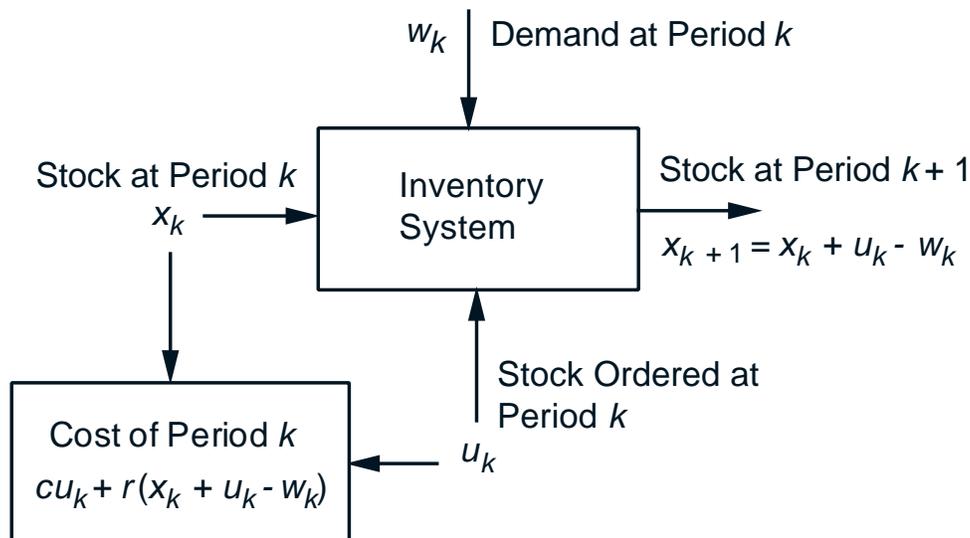
DETERMINISTIC SCHEDULING EXAMPLE

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



- Start from the last tail subproblem and go backwards
- At each state-time pair, we record the optimal cost-to-go and the optimal decision

STOCHASTIC INVENTORY EXAMPLE



- Tail Subproblems of Length 1:

$$J_{N-1}(x_{N-1}) = \min_{u_{N-1} \geq 0} E \left\{ cu_{N-1} + r(x_{N-1} + u_{N-1} - w_{N-1}) \right\}$$

- Tail Subproblems of Length $N - k$:

$$J_k(x_k) = \min_{u_k \geq 0} E \left\{ cu_k + r(x_k + u_k - w_k) + J_{k+1}(x_k + u_k - w_k) \right\}$$

- $J_0(x_0)$ is opt. cost of initial state x_0

DP ALGORITHM

- Start with

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

and go backwards using

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right\}, \quad k = 0, 1, \dots, N-1.$$

- Then $J_0(x_0)$, generated at the last step, is equal to the optimal cost $J^*(x_0)$. Also, the policy

$$\pi^* = \{\mu_0^*, \dots, \mu_{N-1}^*\}$$

where $\mu_k^*(x_k)$ minimizes in the right side above for each x_k and k , is optimal

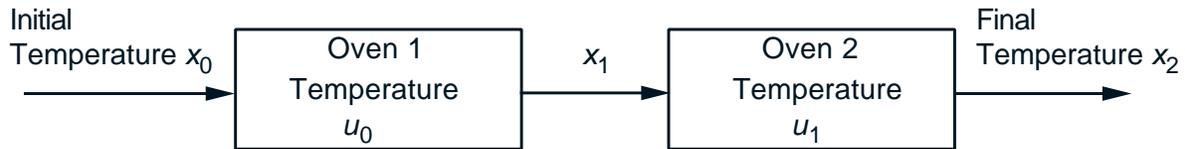
- **Justification:** Proof by induction that $J_k(x_k)$ is equal to $J_k^*(x_k)$, defined as the optimal cost of the tail subproblem that starts at time k at state x_k
- Note:
 - ALL the tail subproblems are solved (in addition to the original problem)
 - Intensive computational requirements

PROOF OF THE INDUCTION STEP

- Let $\pi_k = \{\mu_k, \mu_{k+1}, \dots, \mu_{N-1}\}$ denote a tail policy from time k onward
- Assume that $J_{k+1}(x_{k+1}) = J_{k+1}^*(x_{k+1})$. Then

$$\begin{aligned}
 J_k^*(x_k) &= \min_{(\mu_k, \pi_{k+1})} E_{w_k, \dots, w_{N-1}} \left\{ g_k(x_k, \mu_k(x_k), w_k) \right. \\
 &\quad \left. + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\} \\
 &= \min_{\mu_k} E_{w_k} \left\{ g_k(x_k, \mu_k(x_k), w_k) \right. \\
 &\quad \left. + \min_{\pi_{k+1}} \left[E_{w_{k+1}, \dots, w_{N-1}} \left\{ g_N(x_N) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\} \right] \right\} \\
 &= \min_{\mu_k} E_{w_k} \left\{ g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^*(f_k(x_k, \mu_k(x_k), w_k)) \right\} \\
 &= \min_{\mu_k} E_{w_k} \left\{ g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}(f_k(x_k, \mu_k(x_k), w_k)) \right\} \\
 &= \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right\} \\
 &= J_k(x_k)
 \end{aligned}$$

LINEAR-QUADRATIC ANALYTICAL EXAMPLE



- System

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

where a is given scalar from the interval $(0, 1)$

- Cost

$$r(x_2 - T)^2 + u_0^2 + u_1^2$$

where r is given positive scalar

- DP Algorithm:

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

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

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

STATE AUGMENTATION

- When assumptions of the basic problem are violated (e.g., disturbances are correlated, cost is nonadditive, etc) reformulate/augment the state
- DP algorithm still applies, but the problem gets BIGGER
- **Example:** Time lags

$$x_{k+1} = f_k(x_k, x_{k-1}, u_k, w_k)$$

- Introduce additional state variable $y_k = x_{k-1}$.
New system takes the form

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, y_k, u_k, w_k) \\ x_k \end{pmatrix}$$

View $\tilde{x}_k = (x_k, y_k)$ as the new state.

- DP algorithm for the reformulated problem:

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

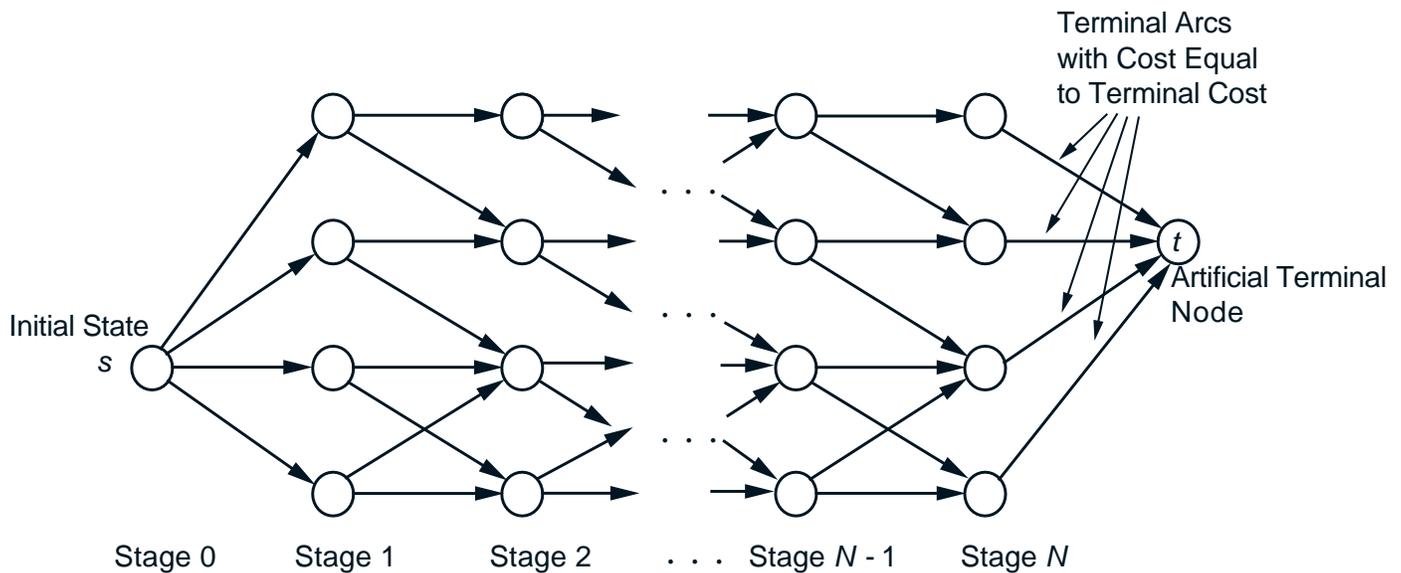
6.231 DYNAMIC PROGRAMMING

LECTURE 3

LECTURE OUTLINE

- Deterministic finite-state DP problems
- Backward shortest path algorithm
- Forward shortest path algorithm
- Shortest path examples
- Alternative shortest path algorithms

DETERMINISTIC FINITE-STATE PROBLEM



- States $\langle == \rangle$ Nodes
- Controls $\langle == \rangle$ Arcs
- Control sequences (open-loop) $\langle == \rangle$ paths from initial state to terminal states
- a_{ij}^k : Cost of transition from state $i \in S_k$ to state $j \in S_{k+1}$ at time k (view it as “length” of the arc)
- a_{it}^N : Terminal cost of state $i \in S_N$
- Cost of control sequence $\langle == \rangle$ Cost of the corresponding path (view it as “length” of the path)

BACKWARD AND FORWARD DP ALGORITHMS

- DP algorithm:

$$J_N(i) = a_{it}^N, \quad i \in S_N,$$

$$J_k(i) = \min_{j \in S_{k+1}} [a_{ij}^k + J_{k+1}(j)], \quad i \in S_k, \quad k = 0, \dots, N-1$$

The optimal cost is $J_0(s)$ and is equal to the length of the shortest path from s to t

- Observation: An optimal path $s \rightarrow t$ is also an optimal path $t \rightarrow s$ in a “reverse” shortest path problem where the direction of each arc is reversed and its length is left unchanged

- Forward DP algorithm (= backward DP algorithm for the reverse problem):

$$\tilde{J}_N(j) = a_{sj}^0, \quad j \in S_1,$$

$$\tilde{J}_k(j) = \min_{i \in S_{N-k}} [a_{ij}^{N-k} + \tilde{J}_{k+1}(i)], \quad j \in S_{N-k+1}$$

The optimal cost is $\tilde{J}_0(t) = \min_{i \in S_N} [a_{it}^N + \tilde{J}_1(i)]$

- View $\tilde{J}_k(j)$ as *optimal cost-to-arrive* to state j from initial state s

A NOTE ON FORWARD DP ALGORITHMS

- There is no forward DP algorithm for **stochastic** problems
- Mathematically, for stochastic problems, we cannot restrict ourselves to open-loop sequences, so the shortest path viewpoint fails
- Conceptually, in the presence of uncertainty, the concept of “optimal-cost-to-arrive” at a state x_k does not make sense. For example, it may be impossible to guarantee (with prob. 1) that any given state can be reached
- By contrast, even in stochastic problems, the concept of “optimal cost-to-go” from any state x_k makes clear sense

GENERIC SHORTEST PATH PROBLEMS

- $\{1, 2, \dots, N, t\}$: nodes of a graph (t : the *destination*)
- a_{ij} : cost of moving from node i to node j
- Find a shortest (minimum cost) path from each node i to node t
- **Assumption: All cycles have nonnegative length.** Then an optimal path need not take more than N moves
- We formulate the problem as one where we require exactly N moves but **allow degenerate moves** from a node i to itself with cost $a_{ii} = 0$

$J_k(i)$ = opt. cost of getting from i to t in $N - k$ moves

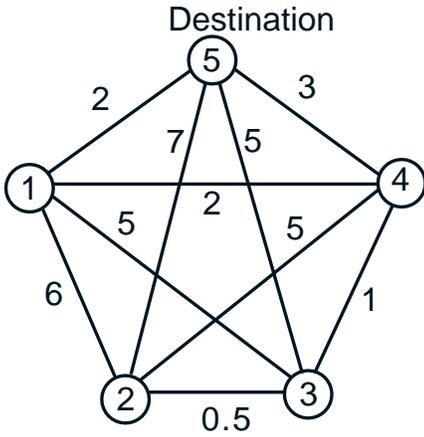
$J_0(i)$: Cost of the optimal path from i to t .

- DP algorithm:

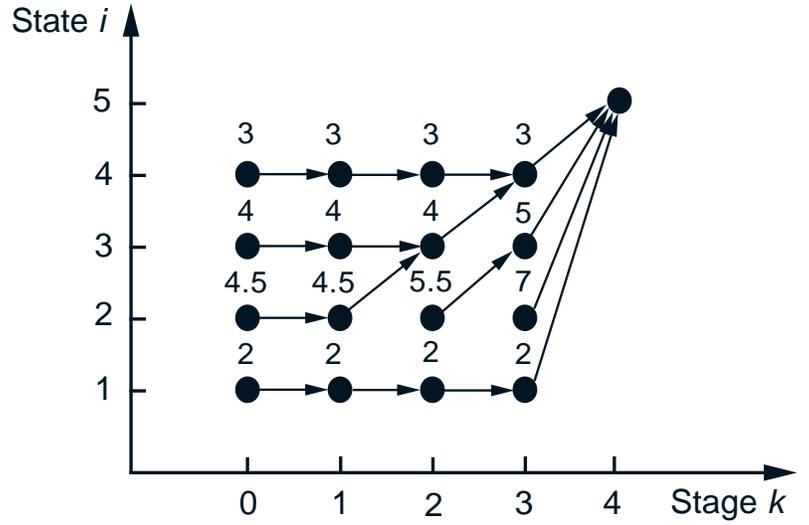
$$J_k(i) = \min_{j=1, \dots, N} [a_{ij} + J_{k+1}(j)], \quad k = 0, 1, \dots, N-2,$$

with $J_{N-1}(i) = a_{it}$, $i = 1, 2, \dots, N$

EXAMPLE



(a)



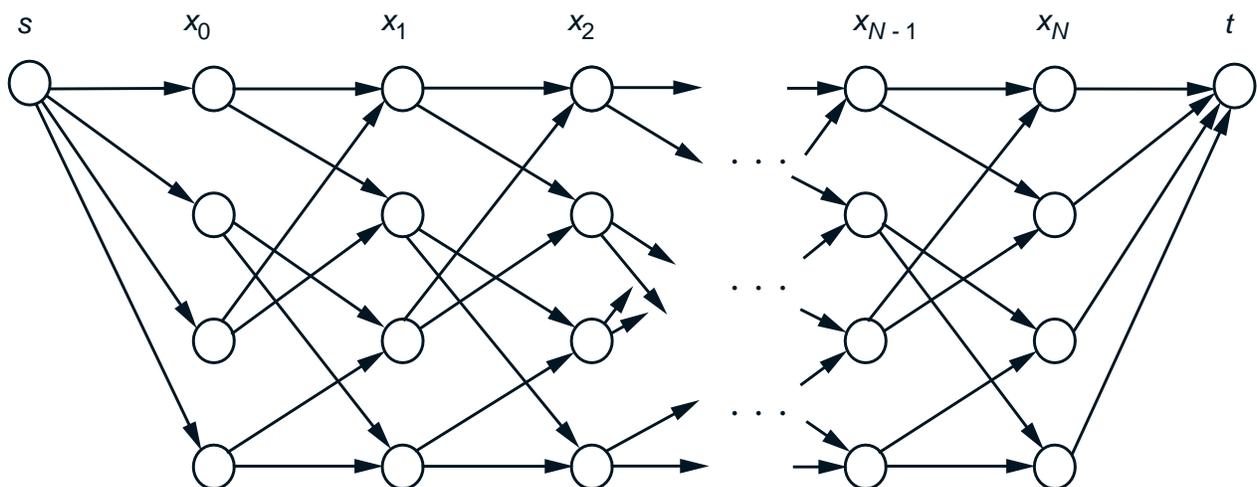
(b)

$$J_{N-1}(i) = a_{it}, \quad i = 1, 2, \dots, N,$$

$$J_k(i) = \min_{j=1, \dots, N} [a_{ij} + J_{k+1}(j)], \quad k = 0, 1, \dots, N-2.$$

ESTIMATION / HIDDEN MARKOV MODELS

- Markov chain with transition probabilities p_{ij}
- State transitions are hidden from view
- For each transition, we get an (independent) observation
- $r(z; i, j)$: Prob. the observation takes value z when the state transition is from i to j
- **Trajectory estimation problem:** Given the observation sequence $Z_N = \{z_1, z_2, \dots, z_N\}$, what is the “most likely” state transition sequence $\hat{X}_N = \{\hat{x}_0, \hat{x}_1, \dots, \hat{x}_N\}$ [one that maximizes $p(X_N | Z_N)$ over all $X_N = \{x_0, x_1, \dots, x_N\}$].



VITERBI ALGORITHM

- We have

$$p(X_N | Z_N) = \frac{p(X_N, Z_N)}{p(Z_N)}$$

where $p(X_N, Z_N)$ and $p(Z_N)$ are the unconditional probabilities of occurrence of (X_N, Z_N) and Z_N

- Maximizing $p(X_N | Z_N)$ is equivalent with maximizing $\ln(p(X_N, Z_N))$

- We have

$$p(X_N, Z_N) = \pi_{x_0} \prod_{k=1}^N p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k)$$

so the problem is equivalent to

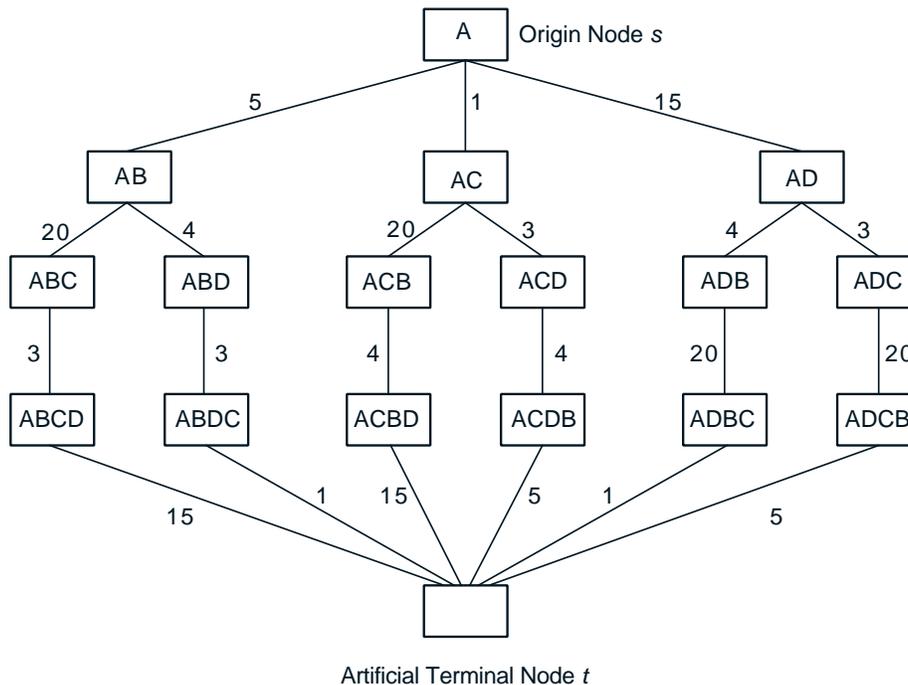
$$\text{minimize } -\ln(\pi_{x_0}) - \sum_{k=1}^N \ln(p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k))$$

over all possible sequences $\{x_0, x_1, \dots, x_N\}$.

- This is a shortest path problem.

GENERAL SHORTEST PATH ALGORITHMS

- There are many nonDP shortest path algorithms. They can all be used to solve deterministic finite-state problems
- They may be preferable than DP if they avoid calculating the optimal cost-to-go of **EVERY** state
- This is essential for problems with **HUGE** state spaces. Such problems arise for example in combinatorial optimization



	5	1	15
5		20	4
1	20		3
15	4	3	

LABEL CORRECTING METHODS

- Given: Origin s , destination t , lengths $a_{ij} \geq 0$.
- Idea is to progressively discover shorter paths from the origin s to every other node i
- **Notation:**
 - d_i (label of i): Length of the shortest path found (initially $d_s = 0$, $d_i = \infty$ for $i \neq s$)
 - UPPER: The label d_t of the destination
 - OPEN list: Contains nodes that are currently active in the sense that they are candidates for further examination (initially $\text{OPEN} = \{s\}$)

Label Correcting Algorithm

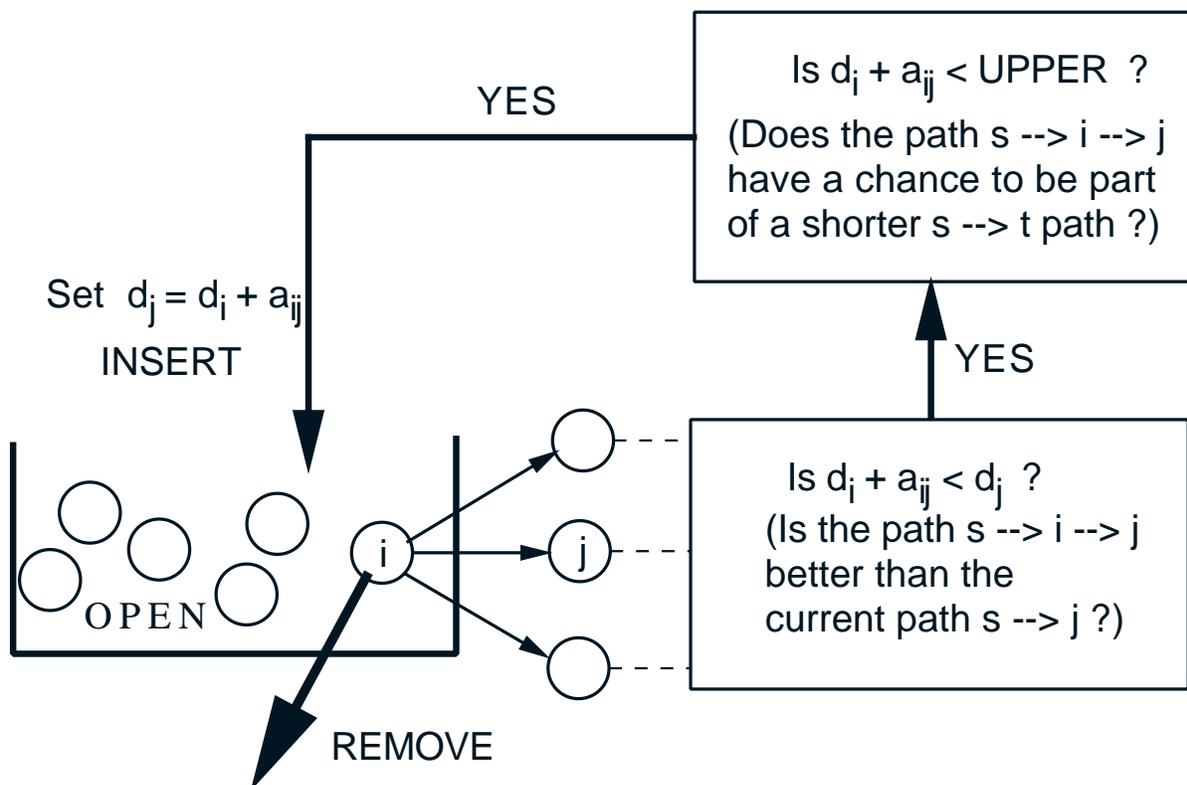
Step 1 (Node Removal): Remove a node i from OPEN and for each child j of i , do step 2

Step 2 (Node Insertion Test): If $d_i + a_{ij} < \min\{d_j, \text{UPPER}\}$, set $d_j = d_i + a_{ij}$ and set i to be the parent of j . In addition, if $j \neq t$, place j in OPEN if it is not already in OPEN, while if $j = t$, set UPPER to the new value $d_i + a_{it}$ of d_t

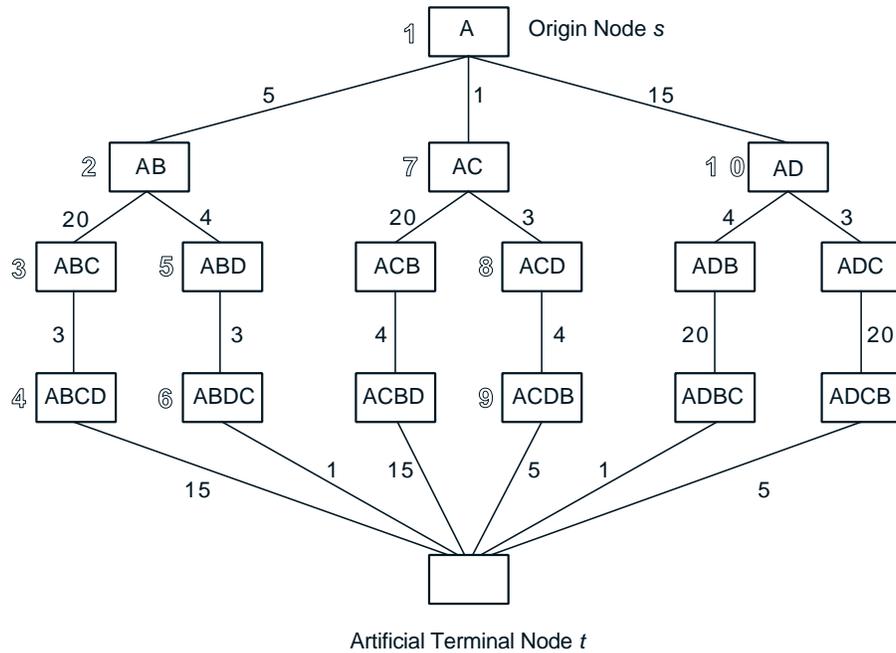
Step 3 (Termination Test): If OPEN is empty, terminate; else go to step 1

VISUALIZATION/EXPLANATION

- Given: Origin s , destination t , lengths $a_{ij} \geq 0$
- d_i (label of i): Length of the shortest path found thus far (initially $d_s = 0$, $d_i = \infty$ for $i \neq s$). The label d_i is implicitly associated with an $s \rightarrow i$ path
- UPPER: The label d_t of the destination
- OPEN list: Contains “active” nodes (initially $\text{OPEN} = \{s\}$)



EXAMPLE



Iter. No.	Node Exiting OPEN	OPEN after Iteration	UPPER
0	-	1	∞
1	1	2, 7, 10	∞
2	2	3, 5, 7, 10	∞
3	3	4, 5, 7, 10	∞
4	4	5, 7, 10	43
5	5	6, 7, 10	43
6	6	7, 10	13
7	7	8, 10	13
8	8	9, 10	13
9	9	10	13
10	10	Empty	13

- Note that **some nodes never entered OPEN**

VALIDITY OF LABEL CORRECTING METHODS

Proposition: If there exists at least one path from the origin to the destination, the label correcting algorithm terminates with UPPER equal to the shortest distance from the origin to the destination

Proof: (1) Each time a node j enters OPEN, its label is decreased and becomes equal to the length of some path from s to j

(2) The number of possible distinct path lengths is finite, so the number of times a node can enter OPEN is finite, and the algorithm terminates

(3) Let $(s, j_1, j_2, \dots, j_k, t)$ be a shortest path and let d^* be the shortest distance. If $\text{UPPER} > d^*$ at termination, UPPER will also be larger than the length of all the paths (s, j_1, \dots, j_m) , $m = 1, \dots, k$, throughout the algorithm. Hence, node j_k will never enter the OPEN list with d_{j_k} equal to the shortest distance from s to j_k . Similarly node j_{k-1} will never enter the OPEN list with $d_{j_{k-1}}$ equal to the shortest distance from s to j_{k-1} . Continue to j_1 to get a contradiction

6.231 DYNAMIC PROGRAMMING

LECTURE 4

LECTURE OUTLINE

- Deterministic continuous-time optimal control
- Examples
- Connection with the calculus of variations
- The Hamilton-Jacobi-Bellman equation as a continuous-time limit of the DP algorithm
- The Hamilton-Jacobi-Bellman equation as a sufficient condition
- Examples

PROBLEM FORMULATION

- **Continuous-time dynamic system:**

$$\dot{x}(t) = f(x(t), u(t)), \quad 0 \leq t \leq T, \quad x(0) : \text{ given,}$$

where

- $x(t) \in \mathfrak{R}^n$: state vector at time t
 - $u(t) \in U \subset \mathfrak{R}^m$: control vector at time t
 - U : control constraint set
 - T : terminal time
- **Admissible control trajectories** $\{u(t) \mid t \in [0, T]\}$:
piecewise continuous functions $\{u(t) \mid t \in [0, T]\}$
with $u(t) \in U$ for all $t \in [0, T]$; uniquely determine
 $\{x(t) \mid t \in [0, T]\}$
 - **Problem:** Find an admissible control trajectory
 $\{u(t) \mid t \in [0, T]\}$ and corresponding state trajectory
 $\{x(t) \mid t \in [0, T]\}$, that minimizes the cost

$$h(x(T)) + \int_0^T g(x(t), u(t)) dt$$

- f, h, g are assumed continuously differentiable

EXAMPLE I

- **Motion control:** A unit mass moves on a line under the influence of a force u
- $x(t) = (x_1(t), x_2(t))$: position and velocity of the mass at time t
- **Problem:** From a given $(x_1(0), x_2(0))$, bring the mass “near” a given final position-velocity pair (\bar{x}_1, \bar{x}_2) at time T in the sense:

$$\text{minimize } |x_1(T) - \bar{x}_1|^2 + |x_2(T) - \bar{x}_2|^2$$

subject to the control constraint

$$|u(t)| \leq 1, \quad \text{for all } t \in [0, T]$$

- The problem fits the framework with

$$\dot{x}_1(t) = x_2(t), \quad \dot{x}_2(t) = u(t),$$

$$h(x(T)) = |x_1(T) - \bar{x}_1|^2 + |x_2(T) - \bar{x}_2|^2,$$

$$g(x(t), u(t)) = 0, \quad \text{for all } t \in [0, T]$$

EXAMPLE II

- A producer with production rate $x(t)$ at time t may allocate a portion $u(t)$ of his/her production rate to reinvestment and $1 - u(t)$ to production of a storable good. Thus $x(t)$ evolves according to

$$\dot{x}(t) = \gamma u(t)x(t),$$

where $\gamma > 0$ is a given constant

- The producer wants to maximize the total amount of product stored

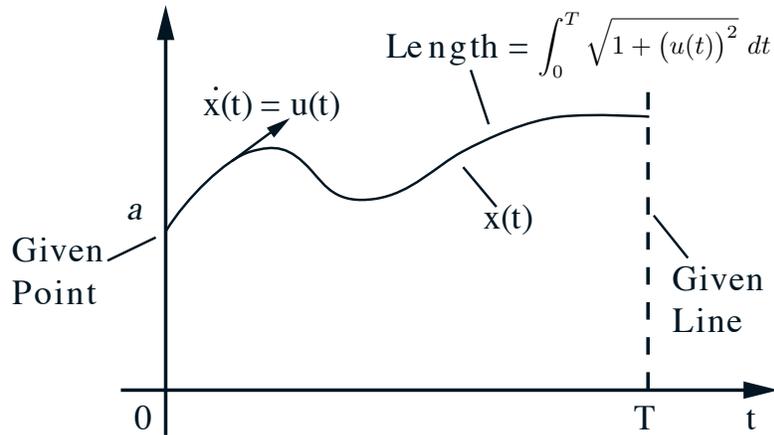
$$\int_0^T (1 - u(t))x(t)dt$$

subject to

$$0 \leq u(t) \leq 1, \quad \text{for all } t \in [0, T]$$

- The initial production rate $x(0)$ is a given positive number

EXAMPLE III (CALCULUS OF VARIATIONS)



- Find a curve from a given point to a given line that has minimum length
- The problem is

$$\begin{aligned} &\text{minimize} \int_0^T \sqrt{1 + (\dot{x}(t))^2} dt \\ &\text{subject to } x(0) = \alpha \end{aligned}$$

- Reformulation as an optimal control problem:

$$\text{minimize} \int_0^T \sqrt{1 + (u(t))^2} dt$$

subject to $\dot{x}(t) = u(t)$, $x(0) = \alpha$

HAMILTON-JACOBI-BELLMAN EQUATION I

- We discretize $[0, T]$ at times $0, \delta, 2\delta, \dots, N\delta$, where $\delta = T/N$, and we let

$$x_k = x(k\delta), \quad u_k = u(k\delta), \quad k = 0, 1, \dots, N$$

- We also discretize the system and cost:

$$x_{k+1} = x_k + f(x_k, u_k) \cdot \delta, \quad h(x_N) + \sum_{k=0}^{N-1} g(x_k, u_k) \cdot \delta$$

- We write the DP algorithm for the discretized problem

$$\tilde{J}^*(N\delta, x) = h(x),$$

$$\tilde{J}^*(k\delta, x) = \min_{u \in U} [g(x, u) \cdot \delta + \tilde{J}^*((k+1)\delta, x + f(x, u) \cdot \delta)].$$

- Assume \tilde{J}^* is differentiable and Taylor-expand:

$$\begin{aligned} \tilde{J}^*(k\delta, x) = \min_{u \in U} [& g(x, u) \cdot \delta + \tilde{J}^*(k\delta, x) + \nabla_t \tilde{J}^*(k\delta, x) \cdot \delta \\ & + \nabla_x \tilde{J}^*(k\delta, x)' f(x, u) \cdot \delta + o(\delta)] \end{aligned}$$

- Cancel $\tilde{J}^*(k\delta, x)$, divide by δ , and take limit

HAMILTON-JACOBI-BELLMAN EQUATION II

- Let $J^*(t, x)$ be the optimal cost-to-go of the continuous problem. Assuming the limit is valid

$$\lim_{k \rightarrow \infty, \delta \rightarrow 0, k\delta = t} \tilde{J}^*(k\delta, x) = J^*(t, x), \quad \text{for all } t, x,$$

we obtain for all t, x ,

$$0 = \min_{u \in U} [g(x, u) + \nabla_t J^*(t, x) + \nabla_x J^*(t, x)' f(x, u)]$$

with the boundary condition $J^*(T, x) = h(x)$

- This is the *Hamilton-Jacobi-Bellman (HJB) equation* – a *partial* differential equation, which is satisfied for all time-state pairs (t, x) by the cost-to-go function $J^*(t, x)$ (assuming J^* is differentiable and the preceding informal limiting procedure is valid)
- Hard to tell *a priori* if $J^*(t, x)$ is differentiable
- So we use the HJB Eq. as a verification tool; if we can solve it for a differentiable $J^*(t, x)$, then:
 - J^* is the optimal-cost-to-go function
 - The control $\mu^*(t, x)$ that minimizes in the RHS for each (t, x) defines an optimal control

VERIFICATION/SUFFICIENCY THEOREM

- Suppose $V(t, x)$ is a solution to the HJB equation; that is, V is continuously differentiable in t and x , and is such that for all t, x ,

$$0 = \min_{u \in U} [g(x, u) + \nabla_t V(t, x) + \nabla_x V(t, x)' f(x, u)],$$

$$V(T, x) = h(x), \quad \text{for all } x$$

- Suppose also that $\mu^*(t, x)$ attains the minimum above for all t and x
- Let $\{x^*(t) \mid t \in [0, T]\}$ and $u^*(t) = \mu^*(t, x^*(t))$, $t \in [0, T]$, be the corresponding state and control trajectories
- Then

$$V(t, x) = J^*(t, x), \quad \text{for all } t, x,$$

and $\{u^*(t) \mid t \in [0, T]\}$ is optimal

- **Limitations of the Theorem**

PROOF

Let $\{(\hat{u}(t), \hat{x}(t)) \mid t \in [0, T]\}$ be any admissible control-state trajectory. We have for all $t \in [0, T]$

$$0 \leq g(\hat{x}(t), \hat{u}(t)) + \nabla_t V(t, \hat{x}(t)) + \nabla_x V(t, \hat{x}(t))' f(\hat{x}(t), \hat{u}(t)).$$

Using the system equation $\dot{\hat{x}}(t) = f(\hat{x}(t), \hat{u}(t))$, the RHS of the above is equal to

$$g(\hat{x}(t), \hat{u}(t)) + \frac{d}{dt} (V(t, \hat{x}(t)))$$

Integrating this expression over $t \in [0, T]$,

$$0 \leq \int_0^T g(\hat{x}(t), \hat{u}(t)) dt + V(T, \hat{x}(T)) - V(0, \hat{x}(0)).$$

Using $V(T, x) = h(x)$ and $\hat{x}(0) = x(0)$, we have

$$V(0, x(0)) \leq h(\hat{x}(T)) + \int_0^T g(\hat{x}(t), \hat{u}(t)) dt.$$

If we use $u^*(t)$ and $x^*(t)$ in place of $\hat{u}(t)$ and $\hat{x}(t)$, the inequalities becomes equalities, and

$$V(0, x(0)) = h(x^*(T)) + \int_0^T g(x^*(t), u^*(t)) dt$$

EXAMPLE OF THE HJB EQUATION

Consider the scalar system $\dot{x}(t) = u(t)$, with $|u(t)| \leq 1$ and cost $(1/2)(x(T))^2$. The HJB equation is

$$0 = \min_{|u| \leq 1} [\nabla_t V(t, x) + \nabla_x V(t, x)u], \quad \text{for all } t, x,$$

with the terminal condition $V(T, x) = (1/2)x^2$

- Evident candidate for optimality: $\mu^*(t, x) = -\text{sgn}(x)$. Corresponding cost-to-go

$$J^*(t, x) = \frac{1}{2} (\max\{0, |x| - (T - t)\})^2.$$

- We verify that J^* solves the HJB Eq., and that $u = -\text{sgn}(x)$ attains the min in the RHS. Indeed,

$$\nabla_t J^*(t, x) = \max\{0, |x| - (T - t)\},$$

$$\nabla_x J^*(t, x) = \text{sgn}(x) \cdot \max\{0, |x| - (T - t)\}.$$

Substituting, the HJB Eq. becomes

$$0 = \min_{|u| \leq 1} [1 + \text{sgn}(x) \cdot u] \max\{0, |x| - (T - t)\}$$

and holds as an identity for all x and t .

LINEAR QUADRATIC PROBLEM

Consider the n -dimensional linear system

$$\dot{x}(t) = Ax(t) + Bu(t),$$

and the quadratic cost

$$x(T)'Q_Tx(T) + \int_0^T (x(t)'Qx(t) + u(t)'Ru(t))dt$$

The HJB equation is

$$0 = \min_{u \in \mathbb{R}^m} [x'Qx + u'Ru + \nabla_t V(t, x) + \nabla_x V(t, x)'(Ax + Bu)],$$

with the terminal condition $V(T, x) = x'Q_Tx$. We try a solution of the form

$$V(t, x) = x'K(t)x, \quad K(t) : n \times n \text{ symmetric,}$$

and show that $V(t, x)$ solves the HJB equation if

$$\dot{K}(t) = -K(t)A - A'K(t) + K(t)BR^{-1}B'K(t) - Q$$

with the terminal condition $K(T) = Q_T$

6.231 DYNAMIC PROGRAMMING

LECTURE 5

LECTURE OUTLINE

- Examples of stochastic DP problems
- Linear-quadratic problems
- Inventory control

LINEAR-QUADRATIC PROBLEMS

- System: $x_{k+1} = A_k x_k + B_k u_k + w_k$
- Quadratic cost

$$E_{w_k, k=0,1,\dots,N-1} \left\{ x'_N Q_N x_N + \sum_{k=0}^{N-1} (x'_k Q_k x_k + u'_k R_k u_k) \right\}$$

where $Q_k \geq 0$ and $R_k > 0$ (in the positive (semi)definite sense).

- w_k are independent and zero mean
- DP algorithm:

$$J_N(x_N) = x'_N Q_N x_N,$$

$$J_k(x_k) = \min_{u_k} E \left\{ x'_k Q_k x_k + u'_k R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k) \right\}$$

- Key facts:
 - $J_k(x_k)$ is quadratic
 - Optimal policy $\{\mu_0^*, \dots, \mu_{N-1}^*\}$ is linear:

$$\mu_k^*(x_k) = L_k x_k$$

- Similar treatment of a number of variants

DERIVATION

- By induction verify that

$$\mu_k^*(x_k) = L_k x_k, \quad J_k(x_k) = x_k' K_k x_k + \text{constant},$$

where L_k are matrices given by

$$L_k = -(B_k' K_{k+1} B_k + R_k)^{-1} B_k' K_{k+1} A_k,$$

and where K_k are symmetric positive semidefinite matrices given by

$$K_N = Q_N,$$

$$K_k = A_k' \left(K_{k+1} - K_{k+1} B_k (B_k' K_{k+1} B_k + R_k)^{-1} B_k' K_{k+1} \right) A_k + Q_k.$$

- This is called the *discrete-time Riccati equation*.
- Just like DP, it starts at the terminal time N and proceeds backwards.
- Certainty equivalence holds (optimal policy is the same as when w_k is replaced by its expected value $E\{w_k\} = 0$).

ASYMPTOTIC BEHAVIOR OF RICCATI EQ.

- Assume time-independent system and cost per stage, and some technical assumptions: controllability of (A, B) and observability of (A, C) where $Q = C'C$
- The Riccati equation converges $\lim_{k \rightarrow -\infty} K_k = K$, where K is pos. definite, and is the unique (within the class of pos. semidefinite matrices) solution of the *algebraic Riccati equation*

$$K = A'(K - KB(B'KB + R)^{-1}B'K)A + Q$$

- The corresponding steady-state controller $\mu^*(x) = Lx$, where

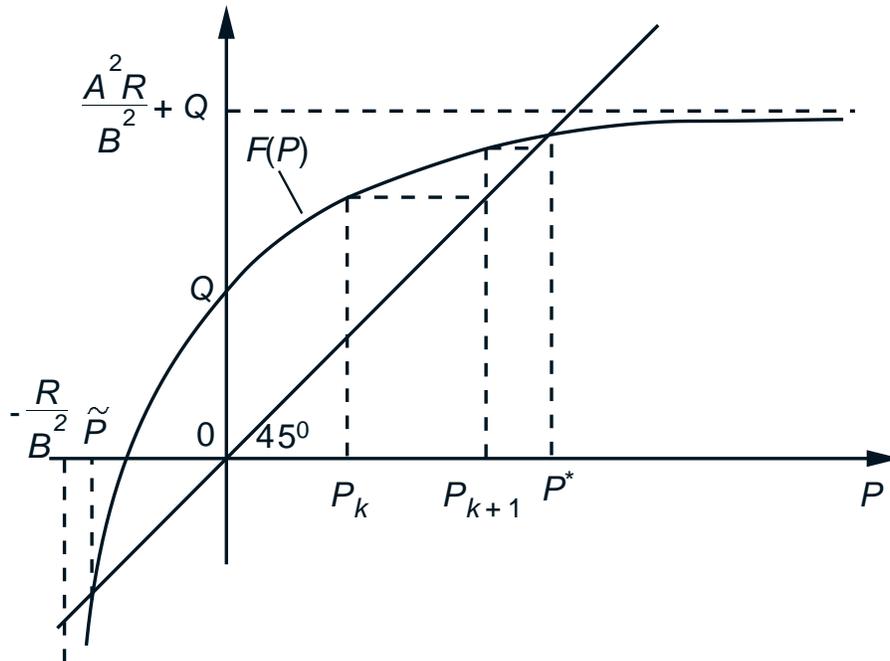
$$L = -(B'KB + R)^{-1}B'KA,$$

is stable in the sense that the matrix $(A + BL)$ of the closed-loop system

$$x_{k+1} = (A + BL)x_k + w_k$$

satisfies $\lim_{k \rightarrow \infty} (A + BL)^k = 0$.

GRAPHICAL PROOF FOR SCALAR SYSTEMS



- Riccati equation (with $P_k = K_{N-k}$):

$$P_{k+1} = A^2 \left(P_k - \frac{B^2 P_k^2}{B^2 P_k + R} \right) + Q,$$

or $P_{k+1} = F(P_k)$, where

$$F(P) = \frac{A^2 R P}{B^2 P + R} + Q.$$

- Note the two steady-state solutions, satisfying $P = F(P)$, of which only one is positive.

RANDOM SYSTEM MATRICES

- Suppose that $\{A_0, B_0\}, \dots, \{A_{N-1}, B_{N-1}\}$ are not known but rather are independent random matrices that are also independent of the w_k
- DP algorithm is

$$J_N(x_N) = x'_N Q_N x_N,$$

$$J_k(x_k) = \min_{u_k} E_{w_k, A_k, B_k} \left\{ x'_k Q_k x_k + u'_k R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k) \right\}$$

- Optimal policy $\mu_k^*(x_k) = L_k x_k$, where

$$L_k = -\left(R_k + E\{B'_k K_{k+1} B_k\}\right)^{-1} E\{B'_k K_{k+1} A_k\},$$

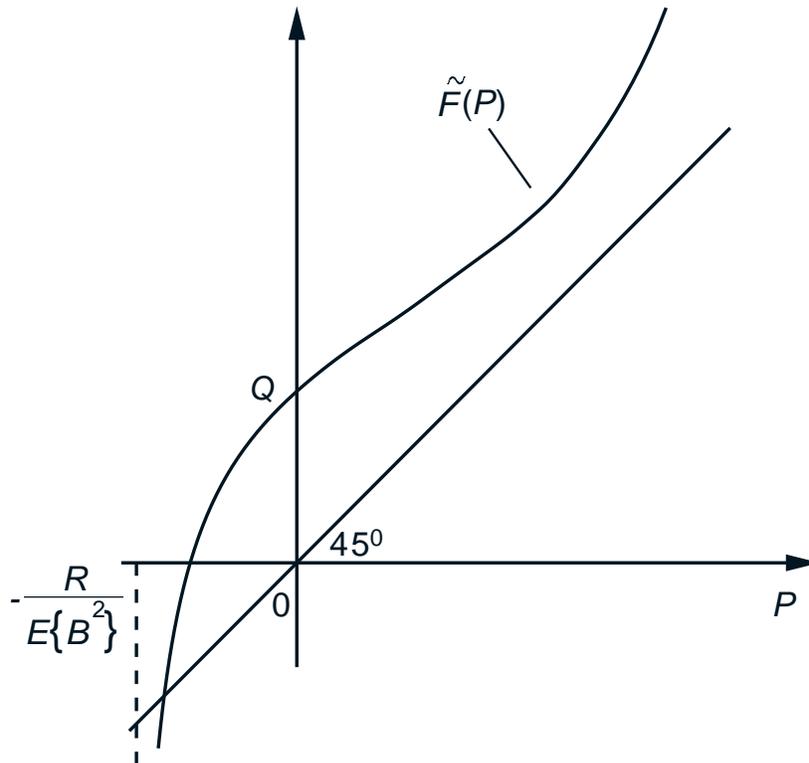
and where the matrices K_k are given by

$$K_N = Q_N,$$

$$K_k = E\{A'_k K_{k+1} A_k\} - E\{A'_k K_{k+1} B_k\} \left(R_k + E\{B'_k K_{k+1} B_k\}\right)^{-1} E\{B'_k K_{k+1} A_k\} + Q_k$$

PROPERTIES

- Certainty equivalence may not hold
- Riccati equation may not converge to a steady-state



- We have $P_{k+1} = \tilde{F}(P_k)$, where

$$\tilde{F}(P) = \frac{E\{A^2\}RP}{E\{B^2\}P + R} + Q + \frac{TP^2}{E\{B^2\}P + R},$$

$$T = E\{A^2\}E\{B^2\} - (E\{A\})^2(E\{B\})^2$$

INVENTORY CONTROL

- x_k : stock, u_k : stock purchased, w_k : demand

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

- Minimize

$$E \left\{ \sum_{k=0}^{N-1} (cu_k + H(x_k + u_k)) \right\}$$

where

$$H(x + u) = E\{r(x + u - w)\}$$

is the expected shortage/holding cost, with r defined e.g., for some $p > 0$ and $h > 0$, as

$$r(x) = p \max(0, -x) + h \max(0, x)$$

- DP algorithm:

$$J_N(x_N) = 0,$$

$$J_k(x_k) = \min_{u_k \geq 0} [cu_k + H(x_k + u_k) + E\{J_{k+1}(x_k + u_k - w_k)\}]$$

OPTIMAL POLICY

- DP algorithm can be written as

$$J_N(x_N) = 0,$$

$$J_k(x_k) = \min_{u_k \geq 0} G_k(x_k + u_k) - cx_k,$$

where

$$G_k(y) = cy + H(y) + E\{J_{k+1}(y - w)\}.$$

- If G_k is convex and $\lim_{|x| \rightarrow \infty} G_k(x) \rightarrow \infty$, we have

$$\mu_k^*(x_k) = \begin{cases} S_k - x_k & \text{if } x_k < S_k, \\ 0 & \text{if } x_k \geq S_k, \end{cases}$$

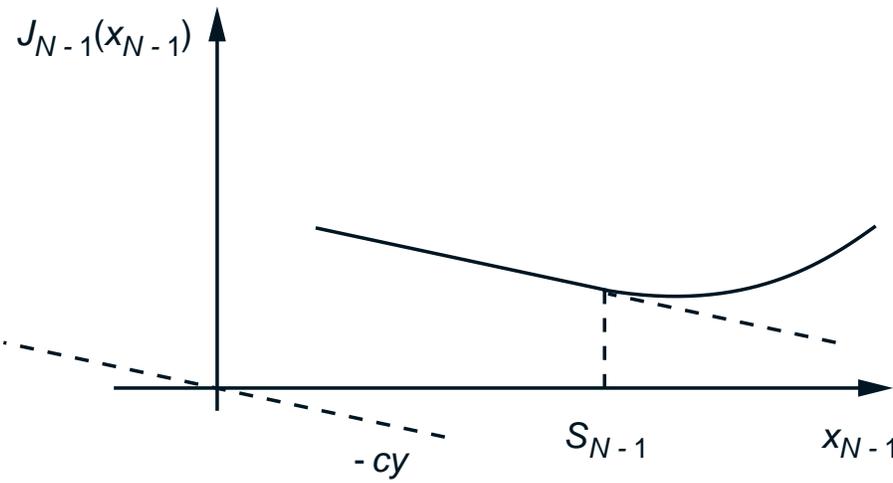
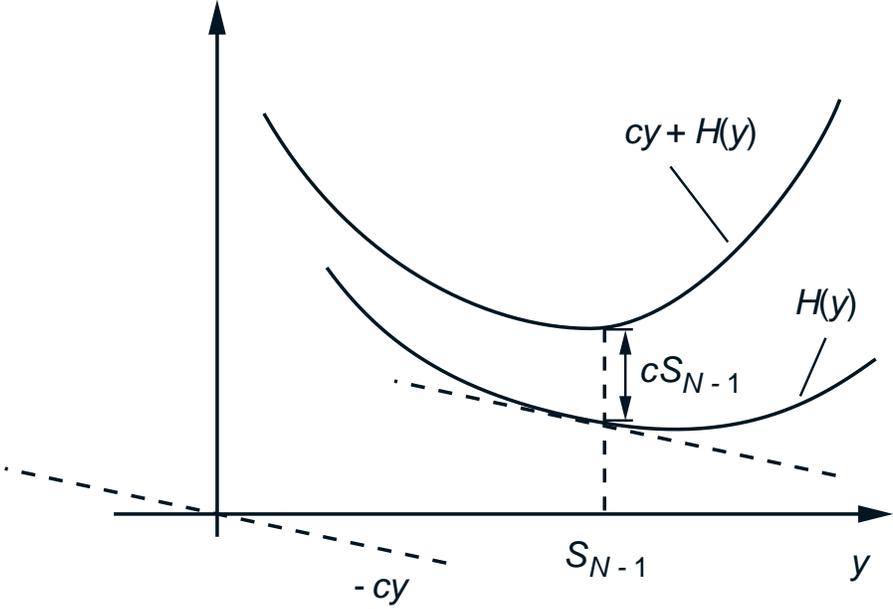
where S_k minimizes $G_k(y)$.

- This is shown, assuming that $c < p$, by showing that J_k is convex for all k , and

$$\lim_{|x| \rightarrow \infty} J_k(x) \rightarrow \infty$$

JUSTIFICATION

- Graphical inductive proof that J_k is convex.



6.231 DYNAMIC PROGRAMMING

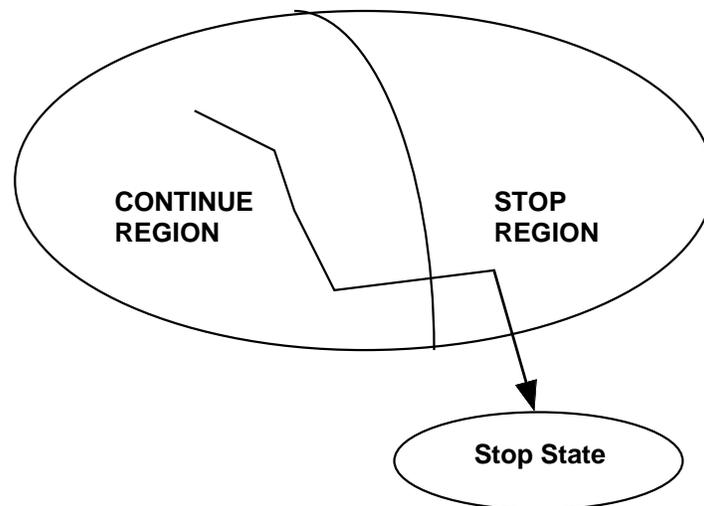
LECTURE 6

LECTURE OUTLINE

- Stopping problems
- Scheduling problems
- Other applications

PURE STOPPING PROBLEMS

- Two possible controls:
 - Stop (incur a one-time stopping cost, and move to cost-free and absorbing stop state)
 - Continue [using $x_{k+1} = f_k(x_k, w_k)$ and incurring the cost-per-stage]
- Each policy consists of a **partition** of the set of states x_k into two regions:
 - **Stop region**, where we stop
 - **Continue region**, where we continue



EXAMPLE: ASSET SELLING

- A person has an asset, and at $k = 0, 1, \dots, N-1$ receives a random offer w_k
- May accept w_k and invest the money at fixed rate of interest r , or reject w_k and wait for w_{k+1} . Must accept the last offer w_{N-1}
- DP algorithm (x_k : current offer, T : stop state):

$$J_N(x_N) = \begin{cases} x_N & \text{if } x_N \neq T, \\ 0 & \text{if } x_N = T, \end{cases}$$

$$J_k(x_k) = \begin{cases} \max\left[(1+r)^{N-k}x_k, E\{J_{k+1}(w_k)\}\right] & \text{if } x_k \neq T, \\ 0 & \text{if } x_k = T. \end{cases}$$

- Optimal policy;

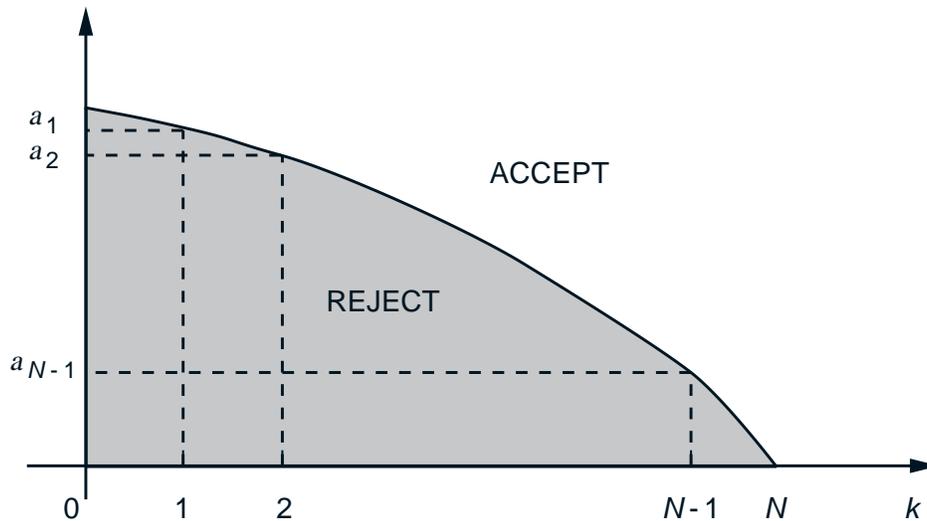
accept the offer x_k if $x_k > \alpha_k$,

reject the offer x_k if $x_k < \alpha_k$,

where

$$\alpha_k = \frac{E\{J_{k+1}(w_k)\}}{(1+r)^{N-k}}.$$

FURTHER ANALYSIS



- Can show that $\alpha_k \geq \alpha_{k+1}$ for all k
- **Proof:** Let $V_k(x_k) = J_k(x_k)/(1+r)^{N-k}$ for $x_k \neq T$. Then the DP algorithm is $V_N(x_N) = x_N$ and

$$V_k(x_k) = \max \left[x_k, (1+r)^{-1} E_w \{ V_{k+1}(w) \} \right].$$

We have $\alpha_k = E_w \{ V_{k+1}(w) \} / (1+r)$, so it is enough to show that $V_k(x) \geq V_{k+1}(x)$ for all x and k . Start with $V_{N-1}(x) \geq V_N(x)$ and use the monotonicity property of DP.

- We can also show that $\alpha_k \rightarrow \bar{a}$ as $k \rightarrow -\infty$. Suggests that for an infinite horizon the optimal policy is stationary.

GENERAL STOPPING PROBLEMS

- At time k , we may stop at cost $t(x_k)$ or choose a control $u_k \in U(x_k)$ and continue

$$J_N(x_N) = t(x_N),$$

$$J_k(x_k) = \min \left[t(x_k), \min_{u_k \in U(x_k)} E \left\{ g(x_k, u_k, w_k) + J_{k+1}(f(x_k, u_k, w_k)) \right\} \right]$$

- Optimal to stop at time k for x in the set

$$T_k = \left\{ x \mid t(x) \leq \min_{u \in U(x)} E \left\{ g(x, u, w) + J_{k+1}(f(x, u, w)) \right\} \right\}$$

- Since $J_{N-1}(x) \leq J_N(x)$, we have $J_k(x) \leq J_{k+1}(x)$ for all k , so

$$T_0 \subset \cdots \subset T_k \subset T_{k+1} \subset \cdots \subset T_{N-1}.$$

- Interesting case is when all the T_k are equal (to T_{N-1} , the set where it is better to stop than to go one step and stop). Can be shown to be true if

$$f(x, u, w) \in T_{N-1}, \quad \text{for all } x \in T_{N-1}, u \in U(x), w.$$

SCHEDULING PROBLEMS

- We have a set of tasks to perform, the ordering is subject to optimal choice.
- Costs depend on the order
- There may be stochastic uncertainty, and precedence and resource availability constraints
- Some of the hardest combinatorial problems are of this type (e.g., traveling salesman, vehicle routing, etc.)
- Some special problems admit a simple quasi-analytical solution method
 - Optimal policy has an “**index form**”, i.e., each task has an easily calculable “cost index”, and it is optimal to select the task that has the minimum value of index (multi-armed bandit problems - to be discussed later)
 - Some problems can be solved by an “**interchange argument**” (start with some schedule, interchange two adjacent tasks, and see what happens). They **require existence of an optimal policy which is open-loop.**

EXAMPLE: THE QUIZ PROBLEM

- Given a list of N questions. If question i is answered correctly (given probability p_i), we receive reward R_i ; if not the quiz terminates. Choose order of questions to maximize expected reward.
- Let i and j be the k th and $(k + 1)$ st questions in an optimally ordered list

$$L = (i_0, \dots, i_{k-1}, i, j, i_{k+2}, \dots, i_{N-1})$$

$$\begin{aligned} E \{ \text{reward of } L \} &= E \{ \text{reward of } \{i_0, \dots, i_{k-1}\} \} \\ &+ p_{i_0} \cdots p_{i_{k-1}} (p_i R_i + p_i p_j R_j) \\ &+ p_{i_0} \cdots p_{i_{k-1}} p_i p_j E \{ \text{reward of } \{i_{k+2}, \dots, i_{N-1}\} \} \end{aligned}$$

Consider the list with i and j interchanged

$$L' = (i_0, \dots, i_{k-1}, j, i, i_{k+2}, \dots, i_{N-1})$$

Since L is optimal, $E\{\text{reward of } L\} \geq E\{\text{reward of } L'\}$, so it follows that $p_i R_i + p_i p_j R_j \geq p_j R_j + p_j p_i R_i$ or

$$p_i R_i / (1 - p_i) \geq p_j R_j / (1 - p_j).$$

MINIMAX CONTROL

- Consider basic problem with the difference that the disturbance w_k instead of being random, it is just known to belong to a given set $W_k(x_k, u_k)$.
- Find policy π that minimizes the cost

$$J_\pi(x_0) = \max_{\substack{w_k \in W_k(x_k, \mu_k(x_k)) \\ k=0,1,\dots,N-1}} \left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right]$$

- The DP algorithm takes the form

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

$$J_k(x_k) = \min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \left[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right]$$

(Exercise 1.5 in the text, solution posted on the [www](#)).

UNKNOWN-BUT-BOUNDED CONTROL

- For each k , keep the x_k of the controlled system

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k)$$

inside a given set X_k , the *target set at time k* .

- This is a minimax control problem, where the cost at stage k is

$$g_k(x_k) = \begin{cases} 0 & \text{if } x_k \in X_k, \\ 1 & \text{if } x_k \notin X_k. \end{cases}$$

- We must reach at time k the set

$$\bar{X}_k = \{x_k \mid J_k(x_k) = 0\}$$

in order to be able to maintain the state within the subsequent target sets.

- Start with $\bar{X}_N = X_N$, and for $k = 0, 1, \dots, N - 1$,

$$\bar{X}_k = \{x_k \in X_k \mid \text{there exists } u_k \in U_k(x_k) \text{ such that } f_k(x_k, u_k, w_k) \in \bar{X}_{k+1}, \text{ for all } w_k \in W_k(x_k, u_k)\}$$

6.231 DYNAMIC PROGRAMMING

LECTURE 7

LECTURE OUTLINE

- Problems with imperfect state info
- Reduction to the perfect state info case
- Linear quadratic problems
- Separation of estimation and control

BASIC PROBL. W/ IMPERFECT STATE INFO

- Same as basic problem of Chapter 1 with one difference: the controller, instead of knowing x_k , receives at each time k an observation of the form

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, u_{k-1}, v_k), \quad k \geq 1$$

- The observation z_k belongs to some space Z_k .
- The random observation disturbance v_k is characterized by a probability distribution

$$P_{v_k}(\cdot \mid x_k, \dots, x_0, u_{k-1}, \dots, u_0, w_{k-1}, \dots, w_0, v_{k-1}, \dots, v_0)$$

- The initial state x_0 is also random and characterized by a probability distribution P_{x_0} .
- The probability distribution $P_{w_k}(\cdot \mid x_k, u_k)$ of w_k is given, and it may depend explicitly on x_k and u_k but not on $w_0, \dots, w_{k-1}, v_0, \dots, v_{k-1}$.
- The control u_k is constrained to a given subset U_k (this subset does not depend on x_k , which is not assumed known).

INFORMATION VECTOR AND POLICIES

- Denote by I_k the **information vector**, i.e., the information available at time k :

$$I_k = (z_0, z_1, \dots, z_k, u_0, u_1, \dots, u_{k-1}), \quad k \geq 1,$$

$$I_0 = z_0$$

- We consider policies $\pi = \{\mu_0, \mu_1, \dots, \mu_{N-1}\}$, where each function μ_k maps the information vector I_k into a control u_k and

$$\mu_k(I_k) \in U_k, \quad \text{for all } I_k, \quad k \geq 0$$

- We want to find a policy π that minimizes

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

subject to the equations

$$x_{k+1} = f_k(x_k, \mu_k(I_k), w_k), \quad k \geq 0,$$

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, \mu_{k-1}(I_{k-1}), v_k), \quad k \geq 1$$

REFORMULATION AS PERFECT INFO PROBL.

- We have

$$I_{k+1} = (I_k, z_{k+1}, u_k), \quad k = 0, 1, \dots, N-2, \quad I_0 = z_0$$

View this as a dynamic system with state I_k , control u_k , and random disturbance z_{k+1}

- We have

$$P(z_{k+1} \mid I_k, u_k) = P(z_{k+1} \mid I_k, u_k, z_0, z_1, \dots, z_k),$$

since z_0, z_1, \dots, z_k are part of the information vector I_k . Thus the probability distribution of z_{k+1} depends explicitly only on the state I_k and control u_k and not on the prior “disturbances” z_k, \dots, z_0

- Write

$$E\{g_k(x_k, u_k, w_k)\} = E\left\{E_{x_k, w_k}\{g_k(x_k, u_k, w_k) \mid I_k, u_k\}\right\}$$

so the cost per stage of the new system is

$$\tilde{g}_k(I_k, u_k) = E_{x_k, w_k}\{g_k(x_k, u_k, w_k) \mid I_k, u_k\}$$

DP ALGORITHM

- Writing the DP algorithm for the (reformulated) perfect state info problem and doing the algebra:

$$J_k(I_k) = \min_{u_k \in U_k} \left[E_{x_k, w_k, z_{k+1}} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \right\} \right]$$

for $k = 0, 1, \dots, N - 2$, and for $k = N - 1$,

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \left[E_{x_{N-1}, w_{N-1}} \left\{ g_N(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})) + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) \mid I_{N-1}, u_{N-1} \right\} \right]$$

- The optimal cost J^* is given by

$$J^* = E_{z_0} \{ J_0(z_0) \}$$

LINEAR-QUADRATIC PROBLEMS

- System: $x_{k+1} = A_k x_k + B_k u_k + w_k$
- Quadratic cost

$$E_{w_k, k=0,1,\dots,N-1} \left\{ x'_N Q_N x_N + \sum_{k=0}^{N-1} (x'_k Q_k x_k + u'_k R_k u_k) \right\}$$

where $Q_k \geq 0$ and $R_k > 0$

- Observations

$$z_k = C_k x_k + v_k, \quad k = 0, 1, \dots, N - 1$$

- $w_0, \dots, w_{N-1}, v_0, \dots, v_{N-1}$ indep. zero mean
- Key fact to show:

– Optimal policy $\{\mu_0^*, \dots, \mu_{N-1}^*\}$ is of the form:

$$\mu_k^*(I_k) = L_k E\{x_k \mid I_k\}$$

L_k : same as for the perfect state info case

- Estimation problem and control problem can be solved separately

DP ALGORITHM I

- Last stage $N - 1$ (suppressing index $N - 1$):

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1}} \left[E_{x_{N-1}, w_{N-1}} \left\{ x'_{N-1} Q x_{N-1} \right. \right. \\ \left. \left. + u'_{N-1} R u_{N-1} + (A x_{N-1} + B u_{N-1} + w_{N-1})' \right. \right. \\ \left. \left. \cdot Q (A x_{N-1} + B u_{N-1} + w_{N-1}) \mid I_{N-1}, u_{N-1} \right\} \right]$$

- Since $E\{w_{N-1} \mid I_{N-1}\} = E\{w_{N-1}\} = 0$, the minimization involves

$$\min_{u_{N-1}} \left[u'_{N-1} (B' Q B + R) u_{N-1} \right. \\ \left. + 2 E\{x_{N-1} \mid I_{N-1}\}' A' Q B u_{N-1} \right]$$

The minimization yields the optimal μ_{N-1}^* :

$$u_{N-1}^* = \mu_{N-1}^*(I_{N-1}) = L_{N-1} E\{x_{N-1} \mid I_{N-1}\}$$

where

$$L_{N-1} = -(B' Q B + R)^{-1} B' Q A$$

DP ALGORITHM II

- Substituting in the DP algorithm

$$\begin{aligned}
 J_{N-1}(I_{N-1}) = & \underset{x_{N-1}}{E} \{x'_{N-1} K_{N-1} x_{N-1} \mid I_{N-1}\} \\
 & + \underset{x_{N-1}}{E} \{ (x_{N-1} - E\{x_{N-1} \mid I_{N-1}\})' \\
 & \quad \cdot P_{N-1} (x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}) \mid I_{N-1} \} \\
 & + \underset{w_{N-1}}{E} \{w'_{N-1} Q_N w_{N-1}\},
 \end{aligned}$$

where the matrices K_{N-1} and P_{N-1} are given by

$$\begin{aligned}
 P_{N-1} = & A'_{N-1} Q_N B_{N-1} (R_{N-1} + B'_{N-1} Q_N B_{N-1})^{-1} \\
 & \quad \cdot B'_{N-1} Q_N A_{N-1},
 \end{aligned}$$

$$K_{N-1} = A'_{N-1} Q_N A_{N-1} - P_{N-1} + Q_{N-1}$$

- Note the structure of J_{N-1} : in addition to the quadratic and constant terms, it involves a quadratic in the estimation error

$$x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}$$

DP ALGORITHM III

- DP equation for period $N - 2$:

$$\begin{aligned}
 J_{N-2}(I_{N-2}) &= \min_{u_{N-2}} \left[\begin{aligned} &E_{x_{N-2}, w_{N-2}, z_{N-1}} \{x'_{N-2} Q x_{N-2} \\ &+ u'_{N-2} R u_{N-2} + J_{N-1}(I_{N-1}) \mid I_{N-2}, u_{N-2}\} \end{aligned} \right] \\
 &= E \{x'_{N-2} Q x_{N-2} \mid I_{N-2}\} \\
 &\quad + \min_{u_{N-2}} \left[\begin{aligned} &u'_{N-2} R u_{N-2} \\ &+ E \{x'_{N-1} K_{N-1} x_{N-1} \mid I_{N-2}, u_{N-2}\} \end{aligned} \right] \\
 &\quad + E \left\{ \left(x_{N-1} - E \{x_{N-1} \mid I_{N-1}\} \right)' \right. \\
 &\quad \quad \cdot P_{N-1} \left(x_{N-1} - E \{x_{N-1} \mid I_{N-1}\} \right) \mid I_{N-2}, u_{N-2} \left. \right\} \\
 &\quad + E_{w_{N-1}} \{w'_{N-1} Q_N w_{N-1}\}
 \end{aligned}$$

- **Key point:** We have excluded the next to last term from the minimization with respect to u_{N-2}
- This term turns out to be independent of u_{N-2}

QUALITY OF ESTIMATION LEMMA

- **Current estimation error is unaffected by past controls:** For every k , there is a function M_k s.t.

$$x_k - E\{x_k \mid I_k\} = M_k(x_0, w_0, \dots, w_{k-1}, v_0, \dots, v_k),$$

independently of the policy being used

- **Consequence:** Using the lemma,

$$x_{N-1} - E\{x_{N-1} \mid I_{N-1}\} = \xi_{N-1},$$

where

ξ_{N-1} : function of $x_0, w_0, \dots, w_{N-2}, v_0, \dots, v_{N-1}$

- Since ξ_{N-1} is independent of u_{N-2} , the conditional expectation of $\xi'_{N-1} P_{N-1} \xi_{N-1}$ satisfies

$$\begin{aligned} E\{\xi'_{N-1} P_{N-1} \xi_{N-1} \mid I_{N-2}, u_{N-2}\} \\ = E\{\xi'_{N-1} P_{N-1} \xi_{N-1} \mid I_{N-2}\} \end{aligned}$$

and is independent of u_{N-2} .

- So minimization in the DP algorithm yields

$$u_{N-2}^* = \mu_{N-2}^*(I_{N-2}) = L_{N-2} E\{x_{N-2} \mid I_{N-2}\}$$

FINAL RESULT

- Continuing similarly (using also the quality of estimation lemma)

$$\mu_k^*(I_k) = L_k E\{x_k \mid I_k\},$$

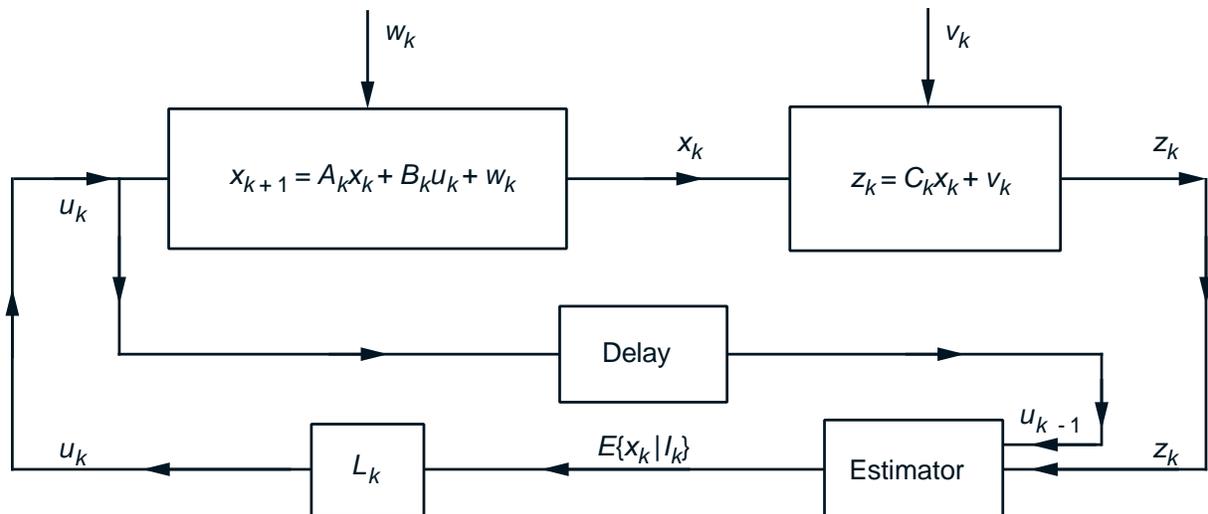
where L_k is the same as for perfect state info:

$$L_k = -(R_k + B_k' K_{k+1} B_k)^{-1} B_k' K_{k+1} A_k,$$

with K_k generated using the Riccati equation:

$$K_N = Q_N, \quad K_k = A_k' K_{k+1} A_k - P_k + Q_k,$$

$$P_k = A_k' K_{k+1} B_k (R_k + B_k' K_{k+1} B_k)^{-1} B_k' K_{k+1} A_k$$



SEPARATION INTERPRETATION

- The optimal controller can be decomposed into
 - (a) An **estimator**, which uses the data to generate the conditional expectation $E\{x_k \mid I_k\}$.
 - (b) An **actuator**, which multiplies $E\{x_k \mid I_k\}$ by the gain matrix L_k and applies the control input $u_k = L_k E\{x_k \mid I_k\}$.
- Generically the estimate \hat{x} of a random vector x given some information (random vector) I , which minimizes the mean squared error

$$E_x\{\|x - \hat{x}\|^2 \mid I\} = \|x\|^2 - 2E\{x \mid I\}\hat{x} + \|\hat{x}\|^2$$

is $E\{x \mid I\}$ (set to zero the derivative with respect to \hat{x} of the above quadratic form).

- The estimator portion of the optimal controller is optimal for the problem of estimating the state x_k assuming the control is not subject to choice.
- The actuator portion is optimal for the control problem assuming perfect state information.

STEADY STATE/IMPLEMENTATION ASPECTS

- As $N \rightarrow \infty$, the solution of the Riccati equation converges to a steady state and $L_k \rightarrow L$.
- If x_0 , w_k , and v_k are Gaussian, $E\{x_k \mid I_k\}$ is a **linear** function of I_k and is generated by a nice recursive algorithm, the Kalman filter.
- The Kalman filter involves also a Riccati equation, so for $N \rightarrow \infty$, and a stationary system, it also has a steady-state structure.
- Thus, for Gaussian uncertainty, the solution is nice and possesses a steady state.
- For nonGaussian uncertainty, computing $E\{x_k \mid I_k\}$ maybe very difficult, so a suboptimal solution is typically used.
- Most common suboptimal controller: Replace $E\{x_k \mid I_k\}$ by the estimate produced by the Kalman filter (act as if x_0 , w_k , and v_k are Gaussian).
- It can be shown that this controller is optimal within the class of controllers that are **linear** functions of I_k .

6.231 DYNAMIC PROGRAMMING

LECTURE 8

LECTURE OUTLINE

- DP for imperfect state info
- Sufficient statistics
- Conditional state distribution as a sufficient statistic
- Finite-state systems
- Examples

REVIEW: IMPERFECT STATE INFO PROBLEM

- Instead of knowing x_k , we receive observations

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, u_{k-1}, v_k), \quad k \geq 0$$

- I_k : information vector available at time k :

$$I_0 = z_0, \quad I_k = (z_0, z_1, \dots, z_k, u_0, u_1, \dots, u_{k-1}), \quad k \geq 1$$

- Optimization over policies $\pi = \{\mu_0, \mu_1, \dots, \mu_{N-1}\}$, where $\mu_k(I_k) \in U_k$, for all I_k and k .
- Find a policy π that minimizes

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

subject to the equations

$$x_{k+1} = f_k(x_k, \mu_k(I_k), w_k), \quad k \geq 0,$$

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, \mu_{k-1}(I_{k-1}), v_k), \quad k \geq 1$$

DP ALGORITHM

- DP algorithm:

$$J_k(I_k) = \min_{u_k \in U_k} \left[E_{x_k, w_k, z_{k+1}} \left\{ g_k(x_k, u_k, w_k) \right. \right. \\ \left. \left. + J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \right\} \right]$$

for $k = 0, 1, \dots, N - 2$, and for $k = N - 1$,

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \left[E_{x_{N-1}, w_{N-1}} \left\{ g_N(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})) \right. \right. \\ \left. \left. + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) \mid I_{N-1}, u_{N-1} \right\} \right]$$

- The optimal cost J^* is given by

$$J^* = E_{z_0} \{ J_0(z_0) \}.$$

SUFFICIENT STATISTICS

- Suppose that we can find a function $S_k(I_k)$ such that the right-hand side of the DP algorithm can be written in terms of some function H_k as

$$\min_{u_k \in U_k} H_k(S_k(I_k), u_k).$$

- Such a function S_k is called a *sufficient statistic*.
- An optimal policy obtained by the preceding minimization can be written as

$$\mu_k^*(I_k) = \bar{\mu}_k(S_k(I_k)),$$

where $\bar{\mu}_k$ is an appropriate function.

- Example of a sufficient statistic: $S_k(I_k) = I_k$
- Another important sufficient statistic

$$S_k(I_k) = P_{x_k|I_k}$$

DP ALGORITHM IN TERMS OF $P_{X_K|I_K}$

- **Filtering Equation:** $P_{x_k|I_k}$ is generated recursively by a dynamic system (estimator) of the form

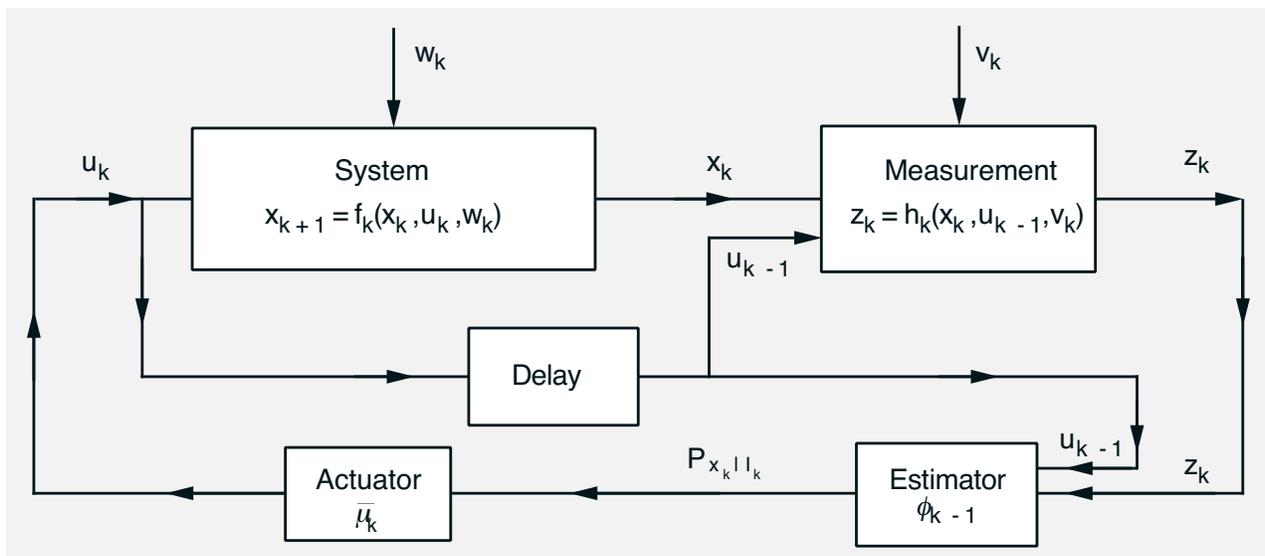
$$P_{x_{k+1}|I_{k+1}} = \Phi_k(P_{x_k|I_k}, u_k, z_{k+1})$$

for a suitable function Φ_k

- DP algorithm can be written as

$$\bar{J}_k(P_{x_k|I_k}) = \min_{u_k \in U_k} \left[E_{x_k, w_k, z_{k+1}} \{ g_k(x_k, u_k, w_k) + \bar{J}_{k+1}(\Phi_k(P_{x_k|I_k}, u_k, z_{k+1})) \mid I_k, u_k \right]$$

- It is the DP algorithm for a **new problem** whose state is $P_{x_k|I_k}$ (also called **belief state**)



EXAMPLE: A SEARCH PROBLEM

- At each period, decide to search or not search a site that may contain a treasure.
- If we search and a treasure is present, we find it with prob. β and remove it from the site.
- Treasure's worth: V . Cost of search: C
- States: treasure present & treasure not present
- Each search can be viewed as an observation of the state
- Denote

p_k : prob. of treasure present at the start of time k
with p_0 given.

- p_k evolves at time k according to the equation

$$p_{k+1} = \begin{cases} p_k & \text{if not search,} \\ 0 & \text{if search and find treasure,} \\ \frac{p_k(1-\beta)}{p_k(1-\beta)+1-p_k} & \text{if search and no treasure.} \end{cases}$$

This is the **filtering equation**.

SEARCH PROBLEM (CONTINUED)

- DP algorithm

$$\bar{J}_k(p_k) = \max \left[0, -C + p_k \beta V \right. \\ \left. + (1 - p_k \beta) \bar{J}_{k+1} \left(\frac{p_k(1 - \beta)}{p_k(1 - \beta) + 1 - p_k} \right) \right],$$

with $\bar{J}_N(p_N) = 0$.

- Can be shown by induction that the functions \bar{J}_k satisfy

$$\bar{J}_k(p_k) \begin{cases} = 0 & \text{if } p_k \leq \frac{C}{\beta V}, \\ > 0 & \text{if } p_k > \frac{C}{\beta V}. \end{cases}$$

- Furthermore, it is optimal to search at period k if and only if

$$p_k \beta V \geq C$$

(expected reward from the next search \geq the cost of the search - a **myopic rule**)

FINITE-STATE SYSTEMS - POMDP

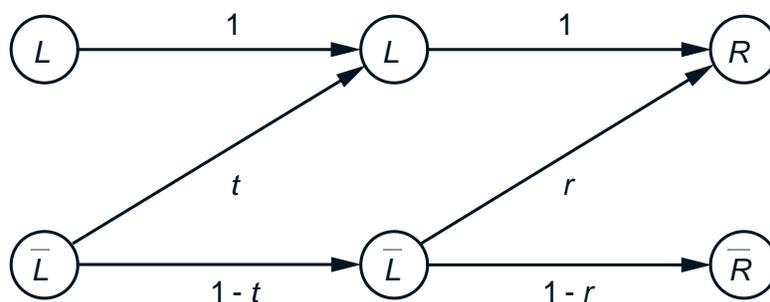
- Suppose the system is a finite-state Markov chain, with states $1, \dots, n$.
- Then the conditional probability distribution $P_{x_k|I_k}$ is an n -vector

$$(P(x_k = 1 | I_k), \dots, P(x_k = n | I_k))$$

- The DP algorithm can be executed over the n -dimensional simplex (state space is not expanding with increasing k)
- When the control and observation spaces are also finite sets the problem is called a POMDP (Partially Observed Markov Decision Problem).
- For POMDP it turns out that the cost-to-go functions \bar{J}_k in the DP algorithm are piecewise linear and concave (Exercise 5.7).
- This is conceptually important. It is also useful in practice because it forms the basis for approximations.

INSTRUCTION EXAMPLE I

- Teaching a student some item. Possible states are L : Item learned, or \bar{L} : Item not learned.
- **Possible decisions:** T : Terminate the instruction, or \bar{T} : Continue the instruction for one period and then conduct a test that indicates whether the student has learned the item.
- **Possible test outcomes:** R : Student gives a correct answer, or \bar{R} : Student gives an incorrect answer.
- Probabilistic structure



- **Cost of instruction:** I per period
- **Cost of terminating instruction:** 0 if student has learned the item, and $C > 0$ if not.

INSTRUCTION EXAMPLE II

- Let p_k : prob. student has learned the item given the test results so far

$$p_k = P(x_k = L \mid z_0, z_1, \dots, z_k).$$

- Using Bayes' rule we obtain the filtering equation

$$\begin{aligned} p_{k+1} &= \Phi(p_k, z_{k+1}) \\ &= \begin{cases} \frac{1-(1-t)(1-p_k)}{1-(1-t)(1-r)(1-p_k)} & \text{if } z_{k+1} = R, \\ 0 & \text{if } z_{k+1} = \bar{R}. \end{cases} \end{aligned}$$

- DP algorithm:

$$\bar{J}_k(p_k) = \min \left[(1 - p_k)C, I + \underset{z_{k+1}}{E} \left\{ \bar{J}_{k+1} \left(\Phi(p_k, z_{k+1}) \right) \right\} \right].$$

starting with

$$\bar{J}_{N-1}(p_{N-1}) = \min \left[(1 - p_{N-1})C, I + (1 - t)(1 - p_{N-1})C \right].$$

INSTRUCTION EXAMPLE III

- Write the DP algorithm as

$$\bar{J}_k(p_k) = \min[(1 - p_k)C, I + A_k(p_k)],$$

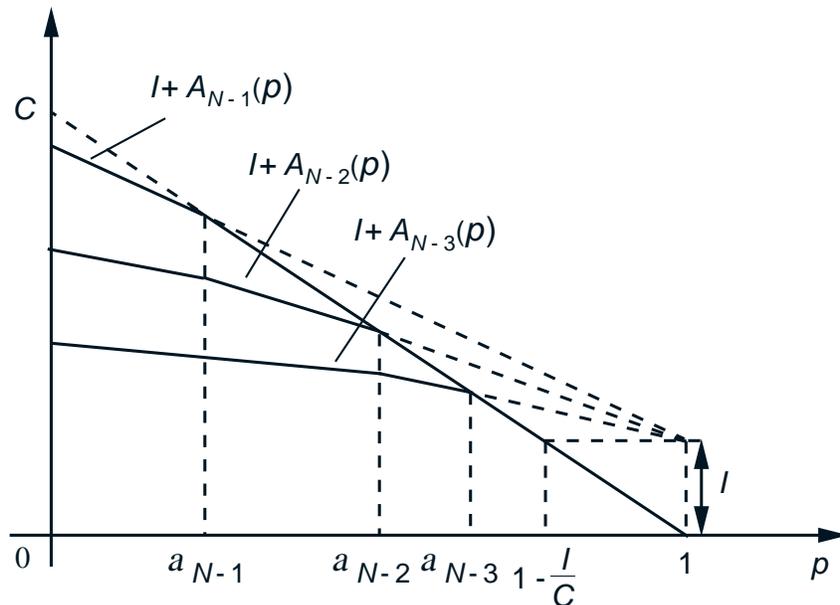
where

$$\begin{aligned} A_k(p_k) = & P(z_{k+1} = R \mid I_k) \bar{J}_{k+1}(\Phi(p_k, R)) \\ & + P(z_{k+1} = \bar{R} \mid I_k) \bar{J}_{k+1}(\Phi(p_k, \bar{R})) \end{aligned}$$

- Can show by induction that $A_k(p)$ are piecewise linear, concave, monotonically decreasing, with

$$A_{k-1}(p) \leq A_k(p) \leq A_{k+1}(p), \quad \text{for all } p \in [0, 1].$$

(The cost-to-go at knowledge prob. p increases as we come closer to the end of horizon.)



6.231 DYNAMIC PROGRAMMING

LECTURE 9

LECTURE OUTLINE

- Suboptimal control
- Cost approximation methods: Classification
- Certainty equivalent control: An example
- Limited lookahead policies
- Performance bounds
- Problem approximation approach
- Parametric cost-to-go approximation

PRACTICAL DIFFICULTIES OF DP

- **The curse of dimensionality**
 - Exponential growth of the computational and storage requirements as the number of state variables and control variables increases
 - Quick explosion of the number of states in combinatorial problems
 - Intractability of imperfect state information problems
- **The curse of modeling**
 - Mathematical models
 - Computer/simulation models
- There may be **real-time solution constraints**
 - A family of problems may be addressed. The data of the problem to be solved is given with little advance notice
 - The problem data may change as the system is controlled – need for on-line replanning

COST-TO-GO FUNCTION APPROXIMATION

- Use a policy computed from the DP equation where **the optimal cost-to-go function J_{k+1} is replaced by an approximation \tilde{J}_{k+1}** . (Sometimes $E\{g_k\}$ is also replaced by an approximation.)

- Apply $\bar{\mu}_k(x_k)$, which attains the minimum in

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

- There are several ways to compute \tilde{J}_{k+1} :
 - **Off-line approximation:** The entire function \tilde{J}_{k+1} is computed for every k , before the control process begins.
 - **On-line approximation:** Only the values $\tilde{J}_{k+1}(x_{k+1})$ at the relevant next states x_{k+1} are computed and used to compute u_k just after the current state x_k becomes known.
 - **Simulation-based methods:** These are off-line and on-line methods that share the common characteristic that they are based on Monte-Carlo simulation. Some of these methods are suitable for problems of very large size.

CERTAINTY EQUIVALENT CONTROL (CEC)

- Idea: **Replace the stochastic problem with a deterministic problem**
- At each time k , the future uncertain quantities are fixed at some “typical” values
- **On-line implementation** for a perfect state info problem. At each time k :

(1) Fix the w_i , $i \geq k$, at some \bar{w}_i . Solve the deterministic problem:

$$\text{minimize } g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \bar{w}_i)$$

where x_k is known, and

$$u_i \in U_i, \quad x_{i+1} = f_i(x_i, u_i, \bar{w}_i).$$

(2) Use as control the first element in the optimal control sequence found.

- So we apply $\bar{\mu}_k(x_k)$ that minimizes

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

where \tilde{J}_{k+1} is the optimal cost of the corresponding deterministic problem.

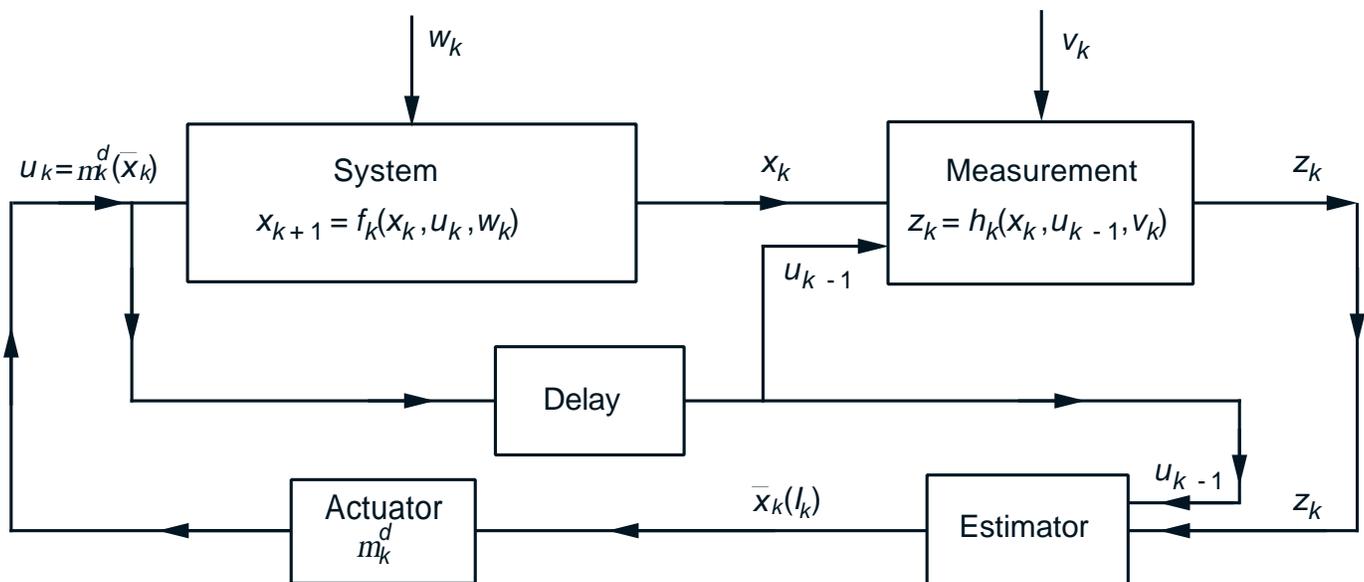
ALTERNATIVE OFF-LINE IMPLEMENTATION

- Let $\{\mu_0^d(x_0), \dots, \mu_{N-1}^d(x_{N-1})\}$ be an optimal controller obtained from the DP algorithm for the deterministic problem

$$\text{minimize } g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), \bar{w}_k)$$

$$\text{subject to } x_{k+1} = f_k(x_k, \mu_k(x_k), \bar{w}_k), \quad \mu_k(x_k) \in U_k$$

- The CEC applies at time k the control input $\mu_k^d(x_k)$.
- In an imperfect info version, x_k is replaced by an estimate $\bar{x}_k(I_k)$.



PARTIALLY STOCHASTIC CEC

- Instead of fixing *all* future disturbances to their typical values, fix only some, and treat the rest as stochastic.
- **Important special case:** Treat an imperfect state information problem as one of perfect state information, using an estimate $\bar{x}_k(I_k)$ of x_k as if it were exact.
- **Multiaccess communication example:** Consider controlling the slotted Aloha system (Example 5.1.1 in the text) by optimally choosing the probability of transmission of waiting packets. This is a hard problem of imperfect state info, whose perfect state info version is easy.
- **Natural partially stochastic CEC:**

$$\tilde{\mu}_k(I_k) = \min \left[1, \frac{1}{\bar{x}_k(I_k)} \right],$$

where $\bar{x}_k(I_k)$ is an estimate of the current packet backlog based on the entire past channel history of successes, idles, and collisions (which is I_k).

GENERAL COST-TO-GO APPROXIMATION

- **One-step lookahead (1SL) policy:** At each k and state x_k , use the control $\bar{\mu}_k(x_k)$ that

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

where

- $\tilde{J}_N = g_N$.
- \tilde{J}_{k+1} : approximation to true cost-to-go J_{k+1}
- **Two-step lookahead policy:** At each k and x_k , use the control $\tilde{\mu}_k(x_k)$ attaining the minimum above, where the function \tilde{J}_{k+1} is obtained using a 1SL approximation (solve a 2-step DP problem).
- If \tilde{J}_{k+1} is readily available and the minimization above is not too hard, the 1SL policy is implementable on-line.
- Sometimes one also replaces $U_k(x_k)$ above with a subset of “most promising controls” $\bar{U}_k(x_k)$.
- As the length of lookahead increases, the required computation quickly explodes.

PERFORMANCE BOUNDS FOR 1SL

- Let $\bar{J}_k(x_k)$ be the cost-to-go from (x_k, k) of the 1SL policy, based on functions \tilde{J}_k .
- Assume that for all (x_k, k) , we have

$$\hat{J}_k(x_k) \leq \tilde{J}_k(x_k), \quad (*)$$

where $\hat{J}_N = g_N$ and for all k ,

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

[so $\hat{J}_k(x_k)$ is computed along with $\bar{\mu}_k(x_k)$]. Then

$$\bar{J}_k(x_k) \leq \hat{J}_k(x_k), \quad \text{for all } (x_k, k).$$

- **Important application:** When \tilde{J}_k is the cost-to-go of some heuristic policy (then the 1SL policy is called the **rollout** policy).
- The bound can be extended to the case where there is a δ_k in the RHS of (*). Then

$$\bar{J}_k(x_k) \leq \tilde{J}_k(x_k) + \delta_k + \cdots + \delta_{N-1}$$

COMPUTATIONAL ASPECTS

- Sometimes nonlinear programming can be used to calculate the 1SL or the multistep version [particularly when $U_k(x_k)$ is not a discrete set]. Connection with **stochastic programming** methods (see text).
- The choice of the approximating functions \tilde{J}_k is critical, and is calculated in a variety of ways.
- Some approaches:
 - (a) **Problem Approximation**: Approximate the optimal cost-to-go with some cost derived from a related but simpler problem
 - (b) **Parametric Cost-to-Go Approximation**: Approximate the optimal cost-to-go with a function of a suitable parametric form, whose parameters are tuned by some heuristic or systematic scheme (Neuro-Dynamic Programming)
 - (c) **Rollout Approach**: Approximate the optimal cost-to-go with the cost of some suboptimal policy, which is calculated either analytically or by simulation

PROBLEM APPROXIMATION

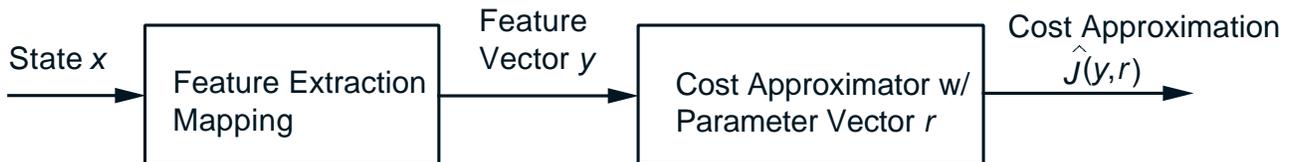
- Many (problem-dependent) possibilities
 - Replace uncertain quantities by nominal values, or simplify the calculation of expected values by limited simulation
 - Simplify difficult constraints or dynamics
- Example of **enforced decomposition**: Route m vehicles that move over a graph. Each node has a “value.” The first vehicle that passes through the node collects its value. Max the total collected value, subject to initial and final time constraints (plus time windows and other constraints).
- Usually the 1-vehicle version of the problem is much simpler. This motivates an approximation obtained by solving single vehicle problems.
- 1SL scheme: At time k and state x_k (position of vehicles and “collected value nodes”), consider all possible k th moves by the vehicles, and at the resulting states we approximate the optimal value-to-go with the value collected **by optimizing the vehicle routes one-at-a-time**

PARAMETRIC COST-TO-GO APPROXIMATION

- Use a cost-to-go approximation from a parametric class $\tilde{J}(x, r)$ where x is the current state and $r = (r_1, \dots, r_m)$ is a vector of “tunable” scalars (weights).
- By adjusting the weights, one can change the “shape” of the approximation \tilde{J} so that it is reasonably close to the true optimal cost-to-go function.
- Two key issues:
 - The **choice of parametric class** $\tilde{J}(x, r)$ (the approximation architecture).
 - Method for **tuning the weights** (“training” the architecture).
- Successful application strongly depends on how these issues are handled, and on insight about the problem.
- Sometimes a simulation-based algorithm is used, particularly when there is no mathematical model of the system.
- We will look in detail at these issues after a few lectures.

APPROXIMATION ARCHITECTURES

- Divided in linear and nonlinear [i.e., linear or nonlinear dependence of $\tilde{J}(x, r)$ on r].
- Linear architectures are easier to train, but nonlinear ones (e.g., neural networks) are richer.
- Architectures based on feature extraction

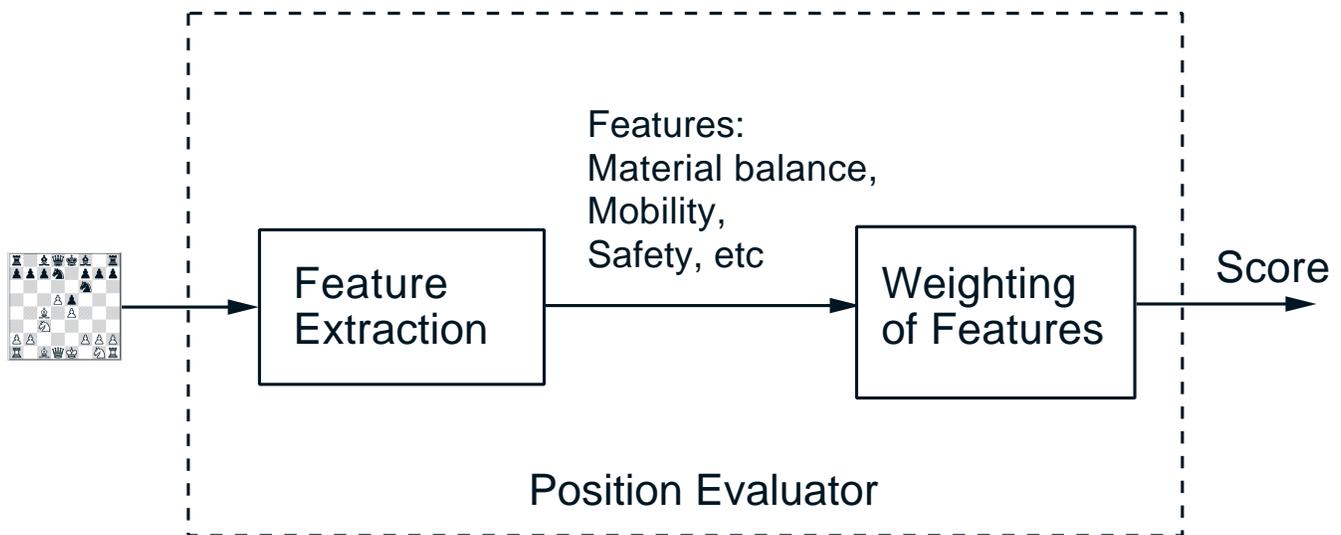


- Ideally, the features will encode much of the nonlinearity that is inherent in the cost-to-go approximated, and the approximation may be quite accurate without a complicated architecture.
- Sometimes the state space is partitioned, and “local” features are introduced for each subset of the partition (they are 0 outside the subset).
- With a well-chosen feature vector $y(x)$, we can use a linear architecture

$$\tilde{J}(x, r) = \hat{J}(y(x), r) = \sum_i r_i y_i(x)$$

AN EXAMPLE - COMPUTER CHESS

- Programs use a feature-based position evaluator that assigns a score to each move/position



- Many context-dependent special features.
- Most often the weighting of features is linear but multistep lookahead is involved.
- Most often the training is done by trial and error.

6.231 DYNAMIC PROGRAMMING

LECTURE 10

LECTURE OUTLINE

- Rollout algorithms
- Cost improvement property
- Discrete deterministic problems
- Approximations of rollout algorithms
- Model Predictive Control (MPC)
- Discretization of continuous time
- Discretization of continuous space
- Other suboptimal approaches

ROLLOUT ALGORITHMS

- **One-step lookahead policy:** At each k and state x_k , use the control $\bar{\mu}_k(x_k)$ that

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

where

- $\tilde{J}_N = g_N$.
- \tilde{J}_{k+1} : approximation to true cost-to-go J_{k+1}
- **Rollout algorithm:** When \tilde{J}_k is the cost-to-go of some heuristic policy (called the *base policy*)
- **Cost improvement property (to be shown):** The rollout algorithm achieves no worse (and usually much better) cost than the base heuristic starting from the same state.
- **Main difficulty:** Calculating $\tilde{J}_k(x_k)$ may be computationally intensive if the cost-to-go of the base policy cannot be analytically calculated.
 - May involve Monte Carlo simulation if the problem is stochastic.
 - Things improve in the deterministic case.

EXAMPLE: THE QUIZ PROBLEM

- A person is given N questions; answering correctly question i has probability p_i , reward v_i . Quiz terminates at the first incorrect answer.
- Problem: Choose the ordering of questions so as to maximize the total expected reward.
- Assuming no other constraints, it is optimal to use the *index policy*: Answer questions in decreasing order of $p_i v_i / (1 - p_i)$.
- With minor changes in the problem, the index policy need not be optimal. Examples:
 - A limit ($< N$) on the maximum number of questions that can be answered.
 - Time windows, sequence-dependent rewards, precedence constraints.
- Rollout with the index policy as base policy: Convenient because at a given state (subset of questions already answered), the index policy and its expected reward can be easily calculated.
- Very effective for solving the quiz problem and important generalizations in scheduling (see Bertsekas and Castanon, J. of Heuristics, Vol. 5, 1999).

COST IMPROVEMENT PROPERTY

- Let

$\bar{J}_k(x_k)$: Cost-to-go of the rollout policy

$H_k(x_k)$: Cost-to-go of the base policy

- We claim that $\bar{J}_k(x_k) \leq H_k(x_k)$ for all x_k, k
- Proof by induction: We have $\bar{J}_N(x_N) = H_N(x_N)$ for all x_N . Assume that

$$\bar{J}_{k+1}(x_{k+1}) \leq H_{k+1}(x_{k+1}), \quad \forall x_{k+1}.$$

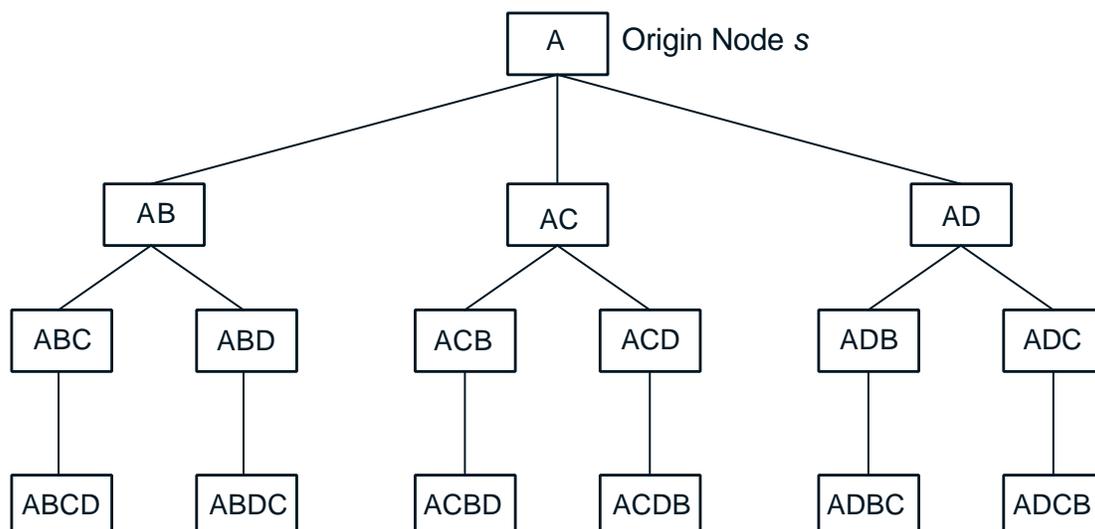
Then, for all x_k

$$\begin{aligned} \bar{J}_k(x_k) &= E \left\{ g_k \left(x_k, \bar{\mu}_k(x_k), w_k \right) + \bar{J}_{k+1} \left(f_k \left(x_k, \bar{\mu}_k(x_k), w_k \right) \right) \right\} \\ &\leq E \left\{ g_k \left(x_k, \bar{\mu}_k(x_k), w_k \right) + H_{k+1} \left(f_k \left(x_k, \bar{\mu}_k(x_k), w_k \right) \right) \right\} \\ &\leq E \left\{ g_k \left(x_k, \mu_k(x_k), w_k \right) + H_{k+1} \left(f_k \left(x_k, \mu_k(x_k), w_k \right) \right) \right\} \\ &= H_k(x_k) \end{aligned}$$

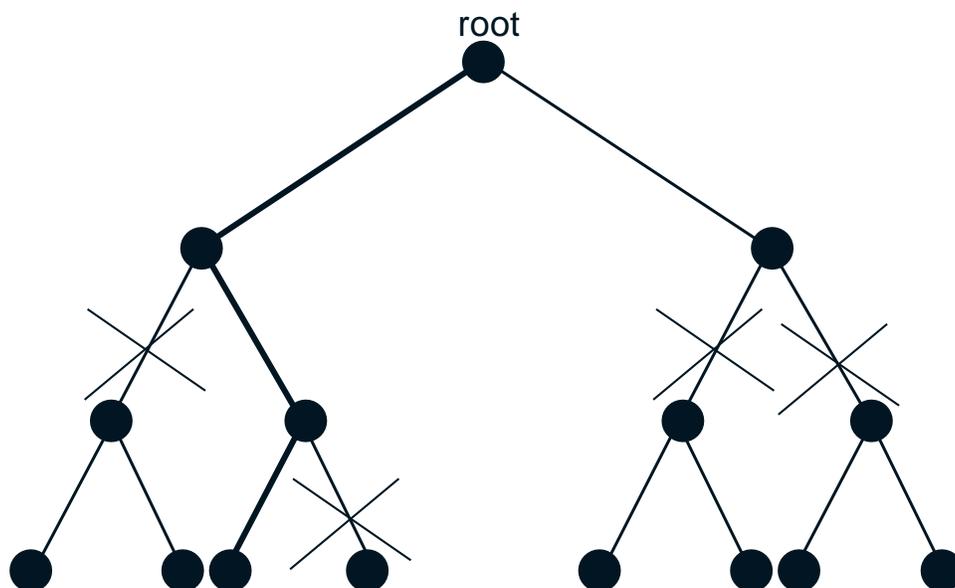
- Induction hypothesis \implies 1st inequality
- Min selection of $\bar{\mu}_k(x_k) \implies$ 2nd inequality
- Definition of $H_k, \mu_k \implies$ last equality

DISCRETE DETERMINISTIC PROBLEMS

- Any discrete optimization problem (with finite number of choices/feasible solutions) can be represented sequentially by breaking down the decision process into stages.
- A tree/shortest path representation. The leaves of the tree correspond to the feasible solutions.
- Decisions can be made in stages.
 - May complete partial solutions, one stage at a time.
 - May apply rollout with any heuristic that can complete a partial solution.
 - No costly stochastic simulation needed.
- **Example:** Traveling salesman problem. Find a minimum cost tour that goes exactly once through each of N cities.



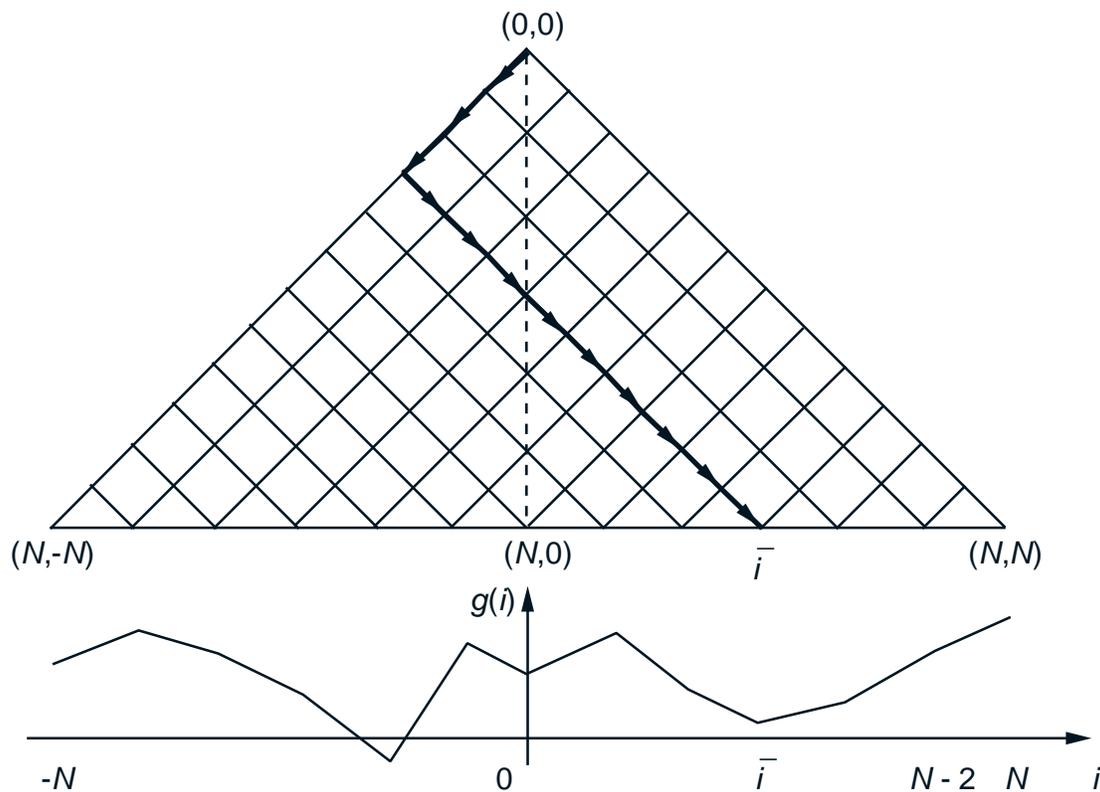
EXAMPLE: THE BREAKTHROUGH PROBLEM



- Given a binary tree with N stages.
- Each arc is either free or is blocked (crossed out in the figure).
- Problem: Find a free path from the root to the leaves (such as the one shown with thick lines).
- **Base heuristic (greedy):** Follow the right branch if free; else follow the left branch if free.
- This is a rare rollout instance that admits a detailed analysis.
- For large N and given prob. of free branch: the rollout algorithm requires $O(N)$ times more computation, but has $O(N)$ times larger prob. of finding a free path than the greedy algorithm.

DET. EXAMPLE: ONE-DIMENSIONAL WALK

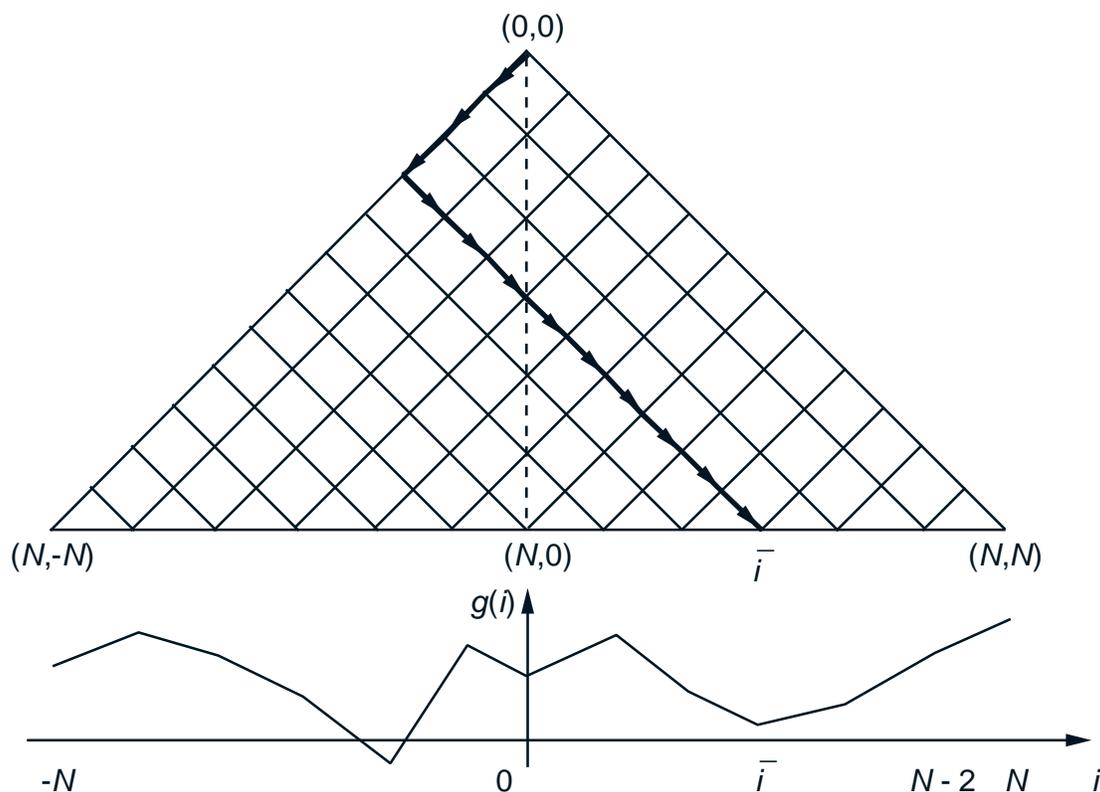
- A person takes either a unit step to the left or a unit step to the right. Minimize the cost $g(i)$ of the point i where he will end up after N steps.



- **Base heuristic:** Always go to the right. Rollout finds the rightmost *local minimum*.
- **Base heuristic:** Compare always go to the right and always go the left. Choose the best of the two. Rollout finds a *global minimum*.

DET. EXAMPLE: ONE-DIMENSIONAL WALK

- A person takes either a unit step to the left or a unit step to the right. Minimize the cost $g(i)$ of the point i where he will end up after N steps.



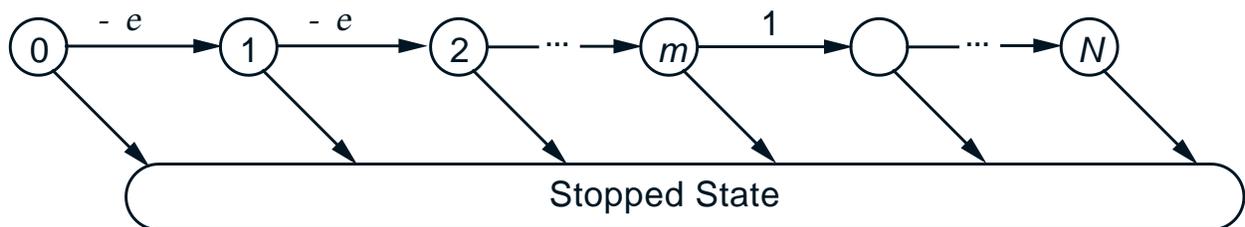
- **Base heuristic:** Always go to the right. Rollout finds the rightmost *local minimum*.
- **Base heuristic:** Compare always go to the right and always go the left. Choose the best of the two. Rollout finds a *global minimum*.

A ROLLOUT ISSUE FOR DISCRETE PROBLEMS

- The base heuristic need not constitute a policy in the DP sense.
- Reason: Depending on its starting point, the base heuristic may not apply the same control at the same state.
- As a result the cost improvement property may be lost (except if the base heuristic has a property called **sequential consistency**; see the text for a formal definition).
- The cost improvement property is restored in two ways:
 - The base heuristic has a property called **sequential improvement** (see the text for a formal definition).
 - A variant of the rollout algorithm, called **fortified rollout**, is used, which enforces cost improvement. Roughly speaking the “best” solution found so far is maintained, and it is followed whenever at any time the standard version of the algorithm tries to follow a “worse” solution (see the text).

ROLLING HORIZON WITH ROLLOUT

- We can use a rolling horizon approximation in calculating the cost-to-go of the base heuristic.
- Because the heuristic is suboptimal, the rationale for a long rolling horizon becomes weaker.
- **Example:** N -stage stopping problem where the stopping cost is 0, the continuation cost is either $-\epsilon$ or 1, where $0 < \epsilon \ll 1$, and the first state with continuation cost equal to 1 is state m . Then the **optimal policy is to stop at state m , and the optimal cost is $-m\epsilon$.**



- Consider the heuristic that continues at every state, and the rollout policy that is based on this heuristic, with a rolling horizon of $l \leq m$ steps.
- It will continue up to the first $m - l + 1$ stages, thus compiling a cost of $-(m - l + 1)\epsilon$. The rollout performance improves as l becomes shorter!
- Limited vision may work to our advantage!

MODEL PREDICTIVE CONTROL (MPC)

- Special case of rollout for linear deterministic systems (similar extensions to nonlinear/stochastic)
 - System: $x_{k+1} = Ax_k + Bu_k$
 - Quadratic cost per stage: $x'_k Q x_k + u'_k R u_k$
 - Constraints: $x_k \in X, u_k \in U(x_k)$
- **Assumption:** For any $x_0 \in X$ there is a feasible state-control sequence that brings the system to 0 in m steps, i.e., $x_m = 0$
- MPC at state x_k solves an m -step optimal control problem with constraint $x_{k+m} = 0$, i.e., finds a sequence $\bar{u}_k, \dots, \bar{u}_{k+m-1}$ that minimizes

$$\sum_{\ell=0}^{m-1} (x'_{k+\ell} Q x_{k+\ell} + u'_{k+\ell} R u_{k+\ell})$$

subject to $x_{k+m} = 0$

- Then applies the first control \bar{u}_k (and repeats at the next state x_{k+1})
- MPC is rollout with heuristic derived from the corresponding $m - 1$ -step optimal control problem
- **Key Property of MPC:** Since the heuristic is stable, the rollout is also stable (suggested by policy improvement property; see the text).

DISCRETIZATION

- If the state space and/or control space is continuous/infinite, it must be discretized.
- Need for consistency, i.e., as the discretization becomes finer, the cost-to-go functions of the discretized problem converge to those of the continuous problem.
- **Pitfall with discretizing continuous time:** The control constraint set may change a lot as we pass to the discrete-time approximation.
- **Example:** Consider the system $\dot{x}(t) = u(t)$, with control constraint $u(t) \in \{-1, 1\}$. The reachable states after time δ are $x(t + \delta) = x(t) + u$, with $u \in [-\delta, \delta]$.
- Compare it with the reachable states after we discretize the system naively: $x(t + \delta) = x(t) + \delta u(t)$, with $u(t) \in \{-1, 1\}$.
- “Convexification effect” of continuous time: a discrete control constraint set in continuous-time differential systems, is equivalent to a continuous control constraint set when the system is looked at discrete times.

SPACE DISCRETIZATION I

- Given a discrete-time system with state space S , consider a finite subset \bar{S} ; for example \bar{S} could be a finite grid within a continuous state space S .
- Difficulty: $f(x, u, w) \notin \bar{S}$ for $x \in \bar{S}$.
- We define **an approximation to the original problem**, with state space \bar{S} , as follows:
- Express each $x \in S$ as a convex combination of states in \bar{S} , i.e.,

$$x = \sum_{x_i \in \bar{S}} \gamma_i(x) x_i \quad \text{where } \gamma_i(x) \geq 0, \quad \sum_i \gamma_i(x) = 1$$

- Define a “reduced” dynamic system with state space \bar{S} , whereby from each $x_i \in \bar{S}$ we move to $\bar{x} = f(x_i, u, w)$ according to the system equation of the original problem, and then move to $x_j \in \bar{S}$ with probabilities $\gamma_j(\bar{x})$.
- Define similarly the corresponding cost per stage of the transitions of the reduced system.
- Note application to finite-state POMDP (Partially Observed Markov Decision Problems)

SPACE DISCRETIZATION II

- Let $\bar{J}_k(x_i)$ be the optimal cost-to-go of the “reduced” problem from each state $x_i \in \bar{S}$ and time k onward.
- Approximate the optimal cost-to-go of any $x \in S$ for the original problem by

$$\tilde{J}_k(x) = \sum_{x_i \in \bar{S}} \gamma_i(x) \bar{J}_k(x_i),$$

and use one-step-lookahead based on \tilde{J}_k .

- The choice of coefficients $\gamma_i(x)$ is in principle arbitrary, but should aim at consistency, i.e., as the number of states in \bar{S} increases, $\tilde{J}_k(x)$ should converge to the optimal cost-to-go of the original problem.
- **Interesting observation:** While the original problem may be deterministic, the reduced problem is always stochastic.
- **Generalization:** The set \bar{S} may be any finite set (not a subset of S) as long as the coefficients $\gamma_i(x)$ admit a meaningful interpretation that quantifies the degree of association of x with x_i .

OTHER SUBOPTIMAL APPROACHES

- **Minimize the DP equation error:** Approximate $J_k(x_k)$ with $\tilde{J}_k(x_k, r_k)$, where r_k is a parameter vector, chosen to minimize some form of error in the DP equations
 - Can be done sequentially going backwards in time (approximate J_k using an approximation of J_{k+1}).

- **Direct approximation of control policies:** For a subset of states x^i , $i = 1, \dots, m$, find

$$\hat{\mu}_k(x^i) = \arg \min_{u_k \in U_k(x^i)} E \left\{ g(x^i, u_k, w_k) + \tilde{J}_{k+1}(f_k(x^i, u_k, w_k), r_{k+1}) \right\}.$$

Then find $\tilde{\mu}_k(x_k, s_k)$, where s_k is a vector of parameters obtained by solving the problem

$$\min_s \sum_{i=1}^m \|\hat{\mu}_k(x^i) - \tilde{\mu}_k(x^i, s)\|^2.$$

- **Approximation in policy space:** Do not bother with cost-to-go approximations. Parametrize the policies as $\tilde{\mu}_k(x_k, s_k)$, and minimize the cost function of the problem over the parameters s_k .

6.231 DYNAMIC PROGRAMMING

LECTURE 11

LECTURE OUTLINE

- Infinite horizon problems
- Stochastic shortest path problems
- Bellman's equation
- Dynamic programming – value iteration
- Discounted problems as special case of SSP

TYPES OF INFINITE HORIZON PROBLEMS

- Same as the basic problem, but:
 - The number of stages is infinite.
 - The system is stationary.
- **Total cost problems:** Minimize

$$J_{\pi}(x_0) = \lim_{N \rightarrow \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}$$

- Stochastic shortest path problems ($\alpha = 1$, finite-state system with a termination state)
 - Discounted problems ($\alpha < 1$, bounded cost per stage)
 - Discounted and undiscounted problems with unbounded cost per stage
- **Average cost problems**

$$\lim_{N \rightarrow \infty} \frac{1}{N} E_{w_k} \left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), w_k) \right\}$$

- Infinite horizon characteristics: Challenging analysis, elegance of solutions and algorithms

PREVIEW OF INFINITE HORIZON RESULTS

- **Key issue:** The relation between the infinite and finite horizon optimal cost-to-go functions.
- **Illustration:** Let $\alpha = 1$ and $J_N(x)$ denote the optimal cost of the N -stage problem, generated after N DP iterations, starting from $J_0(x) \equiv 0$

$$J_{k+1}(x) = \min_{u \in U(x)} \min_w E \{ g(x, u, w) + J_k(f(x, u, w)) \}, \quad \forall x$$

- Typical results for total cost problems:
 - **Convergence of DP algorithm** (value iteration):

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

- **Bellman's equation holds for all x :**

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

- **Optimality condition:** If $\mu(x)$ minimizes in Bellman's Eq., $\{\mu, \mu, \dots\}$ is optimal.
- Bellman's Eq. holds for all types of problems. The other results are true for SSP (and bounded/discounted; unusual exceptions for other problems).

STOCHASTIC SHORTEST PATH PROBLEMS

- Assume finite-state system: States $1, \dots, n$ and special **cost-free termination state t**
 - Transition probabilities $p_{ij}(u)$
 - Control constraints $u \in U(i)$
 - Cost of policy $\pi = \{\mu_0, \mu_1, \dots\}$ is

$$J_\pi(i) = \lim_{N \rightarrow \infty} E \left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k)) \mid x_0 = i \right\}$$

- Optimal policy if $J_\pi(i) = J^*(i)$ for all i .
- Special notation: For stationary policies $\pi = \{\mu, \mu, \dots\}$, we use $J_\mu(i)$ in place of $J_\pi(i)$.
- **Assumption (Termination inevitable):** There exists integer m such that for every policy and initial state, there is positive probability that the termination state will be reached after no more than m stages; for all π , we have

$$\rho_\pi = \max_{i=1, \dots, n} P\{x_m \neq t \mid x_0 = i, \pi\} < 1$$

- Note: We have $\rho = \max_\pi \rho_\pi < 1$, since ρ_π depends only on the first m components of π .

FINITENESS OF POLICY COST FUNCTIONS

- View

$$\rho = \max_{\pi} \rho_{\pi} < 1$$

as an upper bound on the termination prob. over m steps, regardless of policy used.

- For any π and any initial state i

$$\begin{aligned} P\{x_{2m} \neq t \mid x_0 = i, \pi\} &= P\{x_{2m} \neq t \mid x_m \neq t, x_0 = i, \pi\} \\ &\quad \times P\{x_m \neq t \mid x_0 = i, \pi\} \leq \rho^2 \end{aligned}$$

and similarly

$$P\{x_{km} \neq t \mid x_0 = i, \pi\} \leq \rho^k, \quad i = 1, \dots, n$$

- So $E\{\text{Cost between times } km \text{ and } (k+1)m - 1\}$

$$\leq m\rho^k \max_{\substack{i=1, \dots, n \\ u \in U(i)}} |g(i, u)|$$

and

$$|J_{\pi}(i)| \leq \sum_{k=0}^{\infty} m\rho^k \max_{\substack{i=1, \dots, n \\ u \in U(i)}} |g(i, u)| = \frac{m}{1-\rho} \max_{\substack{i=1, \dots, n \\ u \in U(i)}} |g(i, u)|$$

MAIN RESULT

- Given any initial conditions $J_0(1), \dots, J_0(n)$, the sequence $J_k(i)$ generated by the DP iteration

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) J_k(j) \right], \quad \forall i$$

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

- Bellman's equation has $J^*(i)$ as unique solution:

$$J^*(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) J^*(j) \right], \quad \forall i$$

$$J^*(t) = 0$$

- A stationary policy μ is optimal if and only if for every state i , $\mu(i)$ attains the minimum in Bellman's equation.

- **Key proof idea:** The “tail” of the cost series,

$$\sum_{k=mK}^{\infty} E \{ g(x_k, \mu_k(x_k)) \}$$

vanishes as K increases to ∞ .

OUTLINE OF PROOF THAT $J_N \rightarrow J^*$

- Assume for simplicity that $J_0(i) = 0$ for all i . For any $K \geq 1$, write the cost of any policy π as

$$\begin{aligned} J_\pi(x_0) &= \sum_{k=0}^{mK-1} E \left\{ g(x_k, \mu_k(x_k)) \right\} + \sum_{k=mK}^{\infty} E \left\{ g(x_k, \mu_k(x_k)) \right\} \\ &\leq \sum_{k=0}^{mK-1} E \left\{ g(x_k, \mu_k(x_k)) \right\} + \sum_{k=K}^{\infty} \rho^k m \max_{i,u} |g(i, u)| \end{aligned}$$

Take the minimum of both sides over π to obtain

$$J^*(x_0) \leq J_{mK}(x_0) + \frac{\rho^K}{1-\rho} m \max_{i,u} |g(i, u)|.$$

Similarly, we have

$$J_{mK}(x_0) - \frac{\rho^K}{1-\rho} m \max_{i,u} |g(i, u)| \leq J^*(x_0).$$

It follows that $\lim_{K \rightarrow \infty} J_{mK}(x_0) = J^*(x_0)$.

- It can be seen that $J_{mK}(x_0)$ and $J_{mK+k}(x_0)$ converge to the same limit for $k = 1, \dots, m-1$ (since k extra steps far into the future don't matter), so

$$J_N(x_0) \rightarrow J^*(x_0).$$

Similarly, $J_0 \neq 0$ does not matter.

EXAMPLE

- Minimizing the $E\{\text{Time to Termination}\}$: Let

$$g(i, u) = 1, \quad \forall i = 1, \dots, n, \quad u \in U(i)$$

- Under our assumptions, the costs $J^*(i)$ uniquely solve Bellman's equation, which has the form

$$J^*(i) = \min_{u \in U(i)} \left[1 + \sum_{j=1}^n p_{ij}(u) J^*(j) \right], \quad i = 1, \dots, n$$

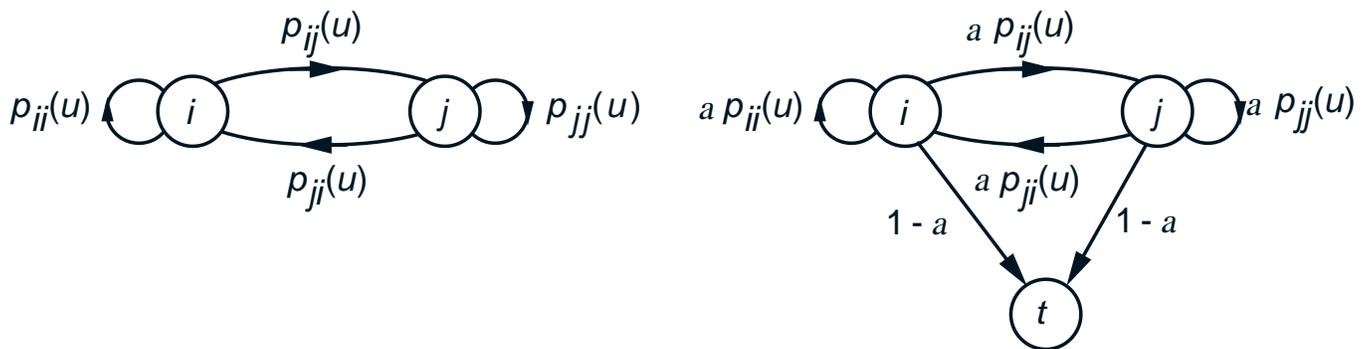
- In the special case where there is only one control at each state, $J^*(i)$ is the mean first passage time from i to t . These times, denoted m_i , are the unique solution of the classical equations

$$m_i = 1 + \sum_{j=1}^n p_{ij} m_j, \quad i = 1, \dots, n,$$

which are seen to be a form of Bellman's equation

DISCOUNTED PROBLEMS

- Assume a discount factor $\alpha < 1$.
- Conversion to an SSP problem.



Policy costs J_π are equal in the two problems.

- Value iteration converges to J^* for all initial J_0 :

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

- J^* is the unique solution of Bellman's equation:

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

- A stationary policy μ is optimal if and only if for every state i , $\mu(i)$ attains the minimum in Bellman's equation.

6.231 DYNAMIC PROGRAMMING

LECTURE 12

LECTURE OUTLINE

- Review of stochastic shortest path problems
- Computational methods for SSP
 - Value iteration
 - Policy iteration
 - Linear programming
- Computational methods for discounted problems

STOCHASTIC SHORTEST PATH PROBLEMS

- Assume finite-state system: States $1, \dots, n$ and special cost-free termination state t
 - Transition probabilities $p_{ij}(u)$
 - Control constraints $u \in U(i)$
 - Cost of policy $\pi = \{\mu_0, \mu_1, \dots\}$ is

$$J_\pi(i) = \lim_{N \rightarrow \infty} E \left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k)) \mid x_0 = i \right\}$$

- Optimal policy if $J_\pi(i) = J^*(i)$ for all i .
- Special notation: For stationary policies $\pi = \{\mu, \mu, \dots\}$, we use $J_\mu(i)$ in place of $J_\pi(i)$.
- **Assumption (Termination inevitable):** There exists integer m such that for every policy and initial state, there is positive probability that the termination state will be reached after no more than m stages; for all π , we have

$$\rho_\pi = \max_{i=1, \dots, n} P\{x_m \neq t \mid x_0 = i, \pi\} < 1$$

MAIN RESULT

- Given any initial conditions $J_0(1), \dots, J_0(n)$, the sequence $J_k(i)$ generated by value iteration

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) J_k(j) \right], \quad \forall i$$

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

- Bellman's equation has $J^*(i)$ as unique solution:

$$J^*(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) J^*(j) \right], \quad \forall i$$

- A stationary policy μ is optimal if and only if for every state i , $\mu(i)$ attains the minimum in Bellman's equation.

- Key proof idea: The “tail” of the cost series,

$$\sum_{k=mK}^{\infty} E \{ g(x_k, \mu_k(x_k)) \}$$

vanishes as K increases to ∞ .

BELLMAN'S EQUATION FOR A SINGLE POLICY

- Consider a stationary policy μ
- $J_\mu(i)$, $i = 1, \dots, n$, are the unique solution of the linear system of n equations

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

- **Proof:** This is just Bellman's equation for a modified/restricted problem where there is only one policy, the stationary policy μ , i.e., the control constraint set at state i is $\tilde{U}(i) = \{\mu(i)\}$
- The equation provides a way to compute $J_\mu(i)$, $i = 1, \dots, n$, but the computation is substantial for large n [$O(n^3)$]
- For large n , value iteration may be preferable. (Typical case of a large linear system of equations, where an iterative method may be better than a direct solution method.)
- For VERY large n , exact methods cannot be applied, and approximations are needed. (We will discuss these later.)

POLICY ITERATION

- It generates a sequence μ^1, μ^2, \dots of stationary policies, starting with any stationary policy μ^0 .
- At the typical iteration, given μ^k , we perform a **policy evaluation step**, that computes the $J_{\mu^k}(i)$ as the solution of the (linear) system of equations

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

in the n unknowns $J(1), \dots, J(n)$. We then perform a **policy improvement step**, which computes a new policy μ^{k+1} as

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) J_{\mu^k}(j) \right], \quad \forall i$$

- The algorithm **terminates when** $J_{\mu^k}(i) = J_{\mu^{k+1}}(i)$ for all i . Then J_{μ^k} satisfies Bellman's Eq. so $J_{\mu^k} = J^*$, and μ^k and μ^{k+1} are optimal.
- Note the **connection with the rollout algorithm**: it is just a single policy iteration.

JUSTIFICATION OF POLICY ITERATION

- We can show that $J_{\mu^{k+1}}(i) \leq J_{\mu^k}(i)$ for all i, k
- Fix k and consider the sequence generated by

$$J_{N+1}(i) = g(i, \mu^{k+1}(i)) + \sum_{j=1}^n p_{ij}(\mu^{k+1}(i)) J_N(j)$$

where $J_0(i) = J_{\mu^k}(i)$. We have

$$\begin{aligned} J_0(i) &= g(i, \mu^k(i)) + \sum_{j=1}^n p_{ij}(\mu^k(i)) J_0(j) \\ &\geq g(i, \mu^{k+1}(i)) + \sum_{j=1}^n p_{ij}(\mu^{k+1}(i)) J_0(j) = J_1(i) \end{aligned}$$

Using the monotonicity property of DP,

$$J_0(i) \geq J_1(i) \geq \cdots \geq J_N(i) \geq J_{N+1}(i) \geq \cdots, \quad \forall i$$

Since $J_N(i) \rightarrow J_{\mu^{k+1}}(i)$ as $N \rightarrow \infty$, we obtain $J_{\mu^k}(i) = J_0(i) \geq J_{\mu^{k+1}}(i)$ for all i . Also if $J_{\mu^k}(i) = J_{\mu^{k+1}}(i)$ for all i , J_{μ^k} solves Bellman's equation and is therefore equal to J^*

- A policy cannot be repeated, there are finitely many stationary policies, so the algorithm terminates with an optimal policy

LINEAR PROGRAMMING

- We claim that J^* is the “largest” J that satisfies the constraint

$$J(i) \leq g(i, u) + \sum_{j=1}^n p_{ij}(u)J(j), \quad (1)$$

for all $i = 1, \dots, n$ and $u \in U(i)$.

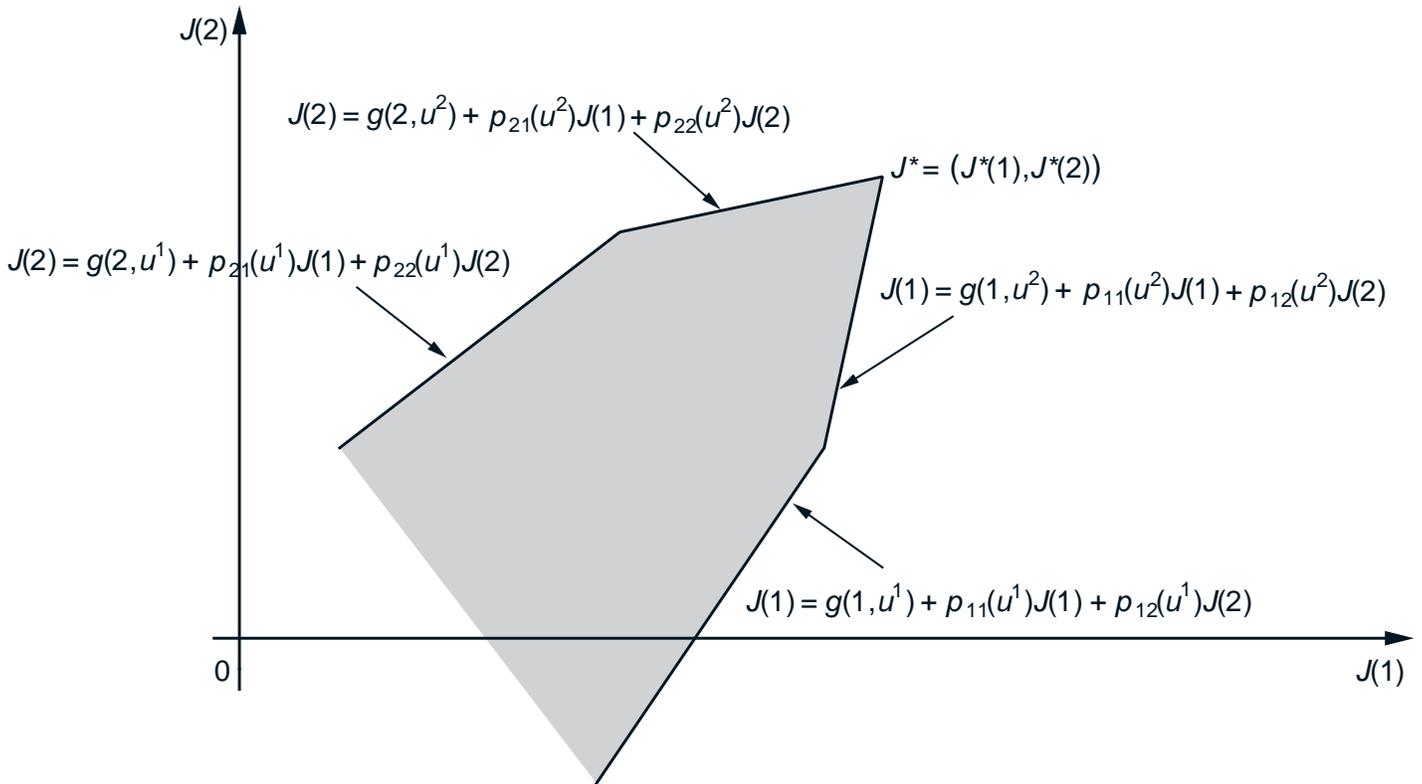
- **Proof:** If we use value iteration to generate a sequence of vectors $J_k = (J_k(1), \dots, J_k(n))$ starting with a J_0 such that

$$J_0(i) \leq \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u)J_0(j) \right], \quad \forall i$$

Then, $J_k(i) \leq J_{k+1}(i)$ for all k and i (monotonicity property of DP) and $J_k \rightarrow J^*$, so that $J_0(i) \leq J^*(i)$ for all i .

- So $J^* = (J^*(1), \dots, J^*(n))$ is the solution of the linear program of maximizing $\sum_{i=1}^n J(i)$ subject to the constraint (1).

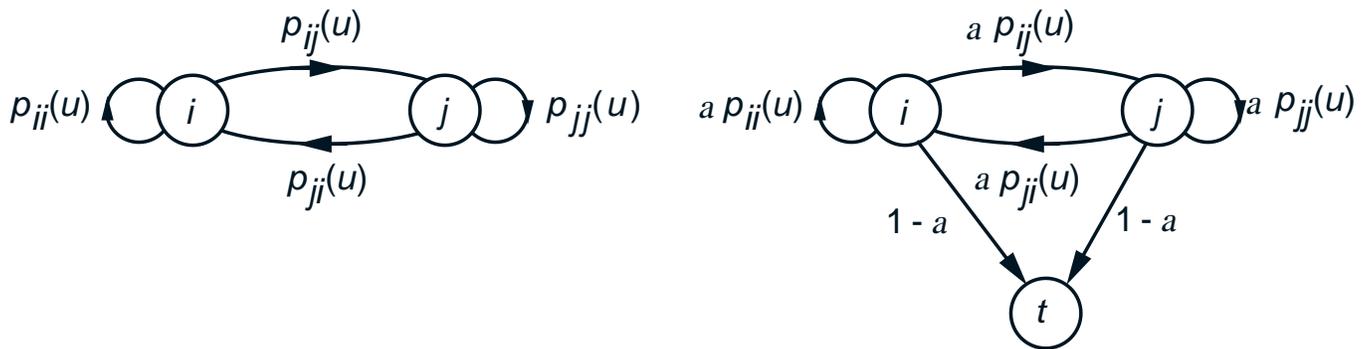
LINEAR PROGRAMMING (CONTINUED)



- Drawback: For large n the dimension of this program is very large. Furthermore, the number of constraints is equal to the number of state-control pairs.

DISCOUNTED PROBLEMS

- Assume a discount factor $\alpha < 1$.
- Conversion to an SSP problem.



- Value iteration converges to J^* for all initial J_0 :

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

- J^* is the unique solution of Bellman's equation:

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

DISCOUNTED PROBLEMS (CONTINUED)

- Policy iteration converges finitely to an optimal policy, and linear programming works.
- **Example:** Asset selling over an infinite horizon. If accepted, the offer x_k of period k , is invested at a rate of interest r .
- By depreciating the sale amount to period 0 dollars, we view $(1 + r)^{-k}x_k$ as the reward for selling the asset in period k at a price x_k , where $r > 0$ is the rate of interest. So the discount factor is $\alpha = 1/(1 + r)$.
- J^* is the unique solution of Bellman's equation

$$J^*(x) = \max \left[x, \frac{E\{J^*(w)\}}{1 + r} \right].$$

- An optimal policy is to sell if and only if the current offer x_k is greater than or equal to $\bar{\alpha}$, where

$$\bar{\alpha} = \frac{E\{J^*(w)\}}{1 + r}.$$

6.231 DYNAMIC PROGRAMMING

LECTURE 13

LECTURE OUTLINE

- Average cost per stage problems
- Connection with stochastic shortest path problems
- Bellman's equation
- Value iteration
- Policy iteration

AVERAGE COST PER STAGE PROBLEM

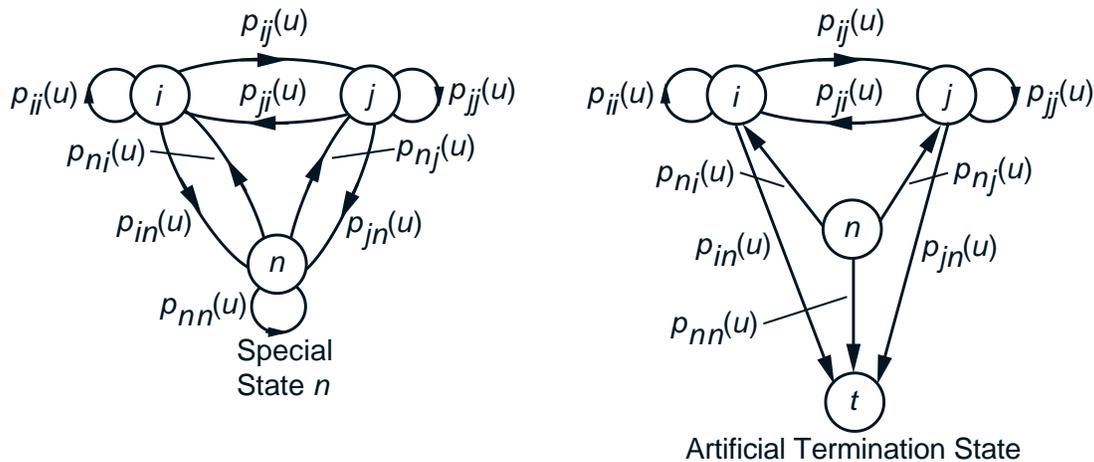
- Assume a stationary system with finite number of states and controls.
- Minimize over policies $\pi = \{\mu_0, \mu_1, \dots\}$

$$J_\pi(x_0) = \lim_{N \rightarrow \infty} \frac{1}{N} \underset{w_k}{E}_{k=0,1,\dots} \left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), w_k) \right\}$$

- Important characteristics (not shared by other types of infinite horizon problems).
 - For any fixed K , **the cost incurred up to time K does not matter** (only the state that we are at time K matters)
 - **If all states “communicate” the optimal cost is independent of initial state** [if we can go from i to j in finite expected time, we must have $J^*(i) \leq J^*(j)$]. So $J^*(i) \equiv \lambda^*$ for all i .
 - Because “communication” issues are so important, the methodology relies heavily on Markov chain theory.
 - **The theory depends a lot on whether the chains corresponding to policies have a single or multiple recurrent classes.** We will focus on the simplest version, using SSP theory.

CONNECTION WITH SSP

- **Assumption:** State n is special, in that for all initial states and all policies, n will be visited infinitely often (with probability 1).
- Divide the sequence of generated states into cycles marked by successive visits to n .
- Each of the cycles can be viewed as a state trajectory of a corresponding stochastic shortest path problem with n as the termination state.



- Let the cost at i of the SSP be $g(i, u) - \lambda^*$
- We will argue that

Av. Cost Probl. \equiv A Min Cost Cycle Probl. \equiv SSP Probl.

CONNECTION WITH SSP (CONTINUED)

- Consider a **minimum cycle cost problem**: Find a stationary policy μ that minimizes the *expected cost per transition within a cycle*

$$\frac{C_{nn}(\mu)}{N_{nn}(\mu)},$$

where for a fixed μ ,

$C_{nn}(\mu) : E\{\text{cost from } n \text{ up to the first return to } n\}$

$N_{nn}(\mu) : E\{\text{time from } n \text{ up to the first return to } n\}$

- Intuitively, $C_{nn}(\mu)/N_{nn}(\mu) =$ average cost of μ , and optimal cycle cost $= \lambda^*$, so

$$C_{nn}(\mu) - N_{nn}(\mu)\lambda^* \geq 0,$$

with equality if μ is optimal.

- Consider **SSP with stage costs** $g(i, u) - \lambda^*$. The cost of μ starting from n is $C_{nn}(\mu) - N_{nn}(\mu)\lambda^*$, so **the optimal/min cycle μ is also optimal for the SSP**.
- Also: **Optimal SSP Cost Starting from $n = 0$** .

BELLMAN'S EQUATION

- Let $h^*(i)$ the optimal cost of this SSP problem when starting at the nontermination states $i = 1, \dots, n$. Then $h^*(1), \dots, h^*(n)$ solve uniquely the corresponding Bellman's equation

$$h^*(i) = \min_{u \in U(i)} \left[g(i, u) - \lambda^* + \sum_{j=1}^{n-1} p_{ij}(u) h^*(j) \right], \forall i$$

- If μ^* is an optimal stationary policy for the SSP problem, we have

$$h^*(n) = C_{nn}(\mu^*) - N_{nn}(\mu^*)\lambda^* = 0$$

- Combining these equations, we have

$$\lambda^* + h^*(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) h^*(j) \right], \forall i$$

$$h^*(n) = 0$$

- If $\mu^*(i)$ attains the min for each i , μ^* is optimal.
- There is also Bellman Eq. for a single policy μ .

MORE ON THE CONNECTION WITH SSP

- Interpretation of $h^*(i)$ as a *relative or differential cost*: It is the minimum of

$E\{\text{cost to reach } n \text{ from } i \text{ for the first time}\}$

– $E\{\text{cost if the stage cost were } \lambda^* \text{ and not } g(i, u)\}$

- We don't know λ^* , so we can't solve the average cost problem as an SSP problem. But similar value and policy iteration algorithms are possible.
- **Example:** A manufacturer at each time:
 - Receives an order with prob. p and no order with prob. $1 - p$.
 - May process all unfilled orders at cost $K > 0$, or process no order at all. The cost per unfilled order at each time is $c > 0$.
 - Maximum number of orders that can remain unfilled is n .
 - Find a processing policy that minimizes the total expected cost per stage.

EXAMPLE (CONTINUED)

- State = number of unfilled orders. State 0 is the special state for the SSP formulation.

- **Bellman's equation:** For states $i = 0, 1, \dots, n-1$

$$\lambda^* + h^*(i) = \min \left[K + (1 - p)h^*(0) + ph^*(1), \right. \\ \left. ci + (1 - p)h^*(i) + ph^*(i + 1) \right],$$

and for state n

$$\lambda^* + h^*(n) = K + (1 - p)h^*(0) + ph^*(1)$$

- **Optimal policy:** Process i unfilled orders if

$$K + (1 - p)h^*(0) + ph^*(1) \leq ci + (1 - p)h^*(i) + ph^*(i + 1).$$

- Intuitively, $h^*(i)$ is monotonically nondecreasing with i (interpret $h^*(i)$ as optimal costs-to-go for the associate SSP problem). So a *threshold policy* is optimal: process the orders if their number exceeds some threshold integer m^* .

VALUE ITERATION

- **Natural value iteration method:** Generate optimal k -stage costs by DP algorithm starting with any J_0 :

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) J_k(j) \right], \quad \forall i$$

- **Result:** $\lim_{k \rightarrow \infty} J_k(i)/k = \lambda^*$ for all i .
- **Proof outline:** Let J_k^* be so generated from the initial condition $J_0^* = h^*$. Then, by induction,

$$J_k^*(i) = k\lambda^* + h^*(i), \quad \forall i, \forall k.$$

On the other hand,

$$|J_k(i) - J_k^*(i)| \leq \max_{j=1, \dots, n} |J_0(j) - h^*(j)|, \quad \forall i$$

since $J_k(i)$ and $J_k^*(i)$ are optimal costs for two k -stage problems that differ only in the terminal cost functions, which are J_0 and h^* .

RELATIVE VALUE ITERATION

- The value iteration method just described has two drawbacks:
 - Since typically some components of J_k diverge to ∞ or $-\infty$, calculating $\lim_{k \rightarrow \infty} J_k(i)/k$ is numerically cumbersome.
 - The method will not compute a corresponding differential cost vector h^* .
- We can bypass both difficulties by subtracting a constant from all components of the vector J_k , so that the difference, call it h_k , remains bounded.
- **Relative value iteration algorithm:** Pick any state s , and iterate according to

$$h_{k+1}(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) h_k(j) \right]$$
$$- \min_{u \in U(s)} \left[g(s, u) + \sum_{j=1}^n p_{sj}(u) h_k(j) \right], \quad \forall i$$

- Then we can show $h_k \rightarrow h^*$ (under an extra assumption).

POLICY ITERATION

- At the typical iteration, we have a stationary μ^k .

- **Policy evaluation:** Compute λ^k and $h^k(i)$ of μ^k , using the $n + 1$ equations $h^k(n) = 0$ and

$$\lambda^k + h^k(i) = g(i, \mu^k(i)) + \sum_{j=1}^n p_{ij}(\mu^k(i)) h^k(j), \quad \forall i$$

- **Policy improvement:** (For the λ^k -SSP) Find for all i

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^n p_{ij}(u) h^k(j) \right]$$

- If $\lambda^{k+1} = \lambda^k$ and $h^{k+1}(i) = h^k(i)$ for all i , stop; otherwise, repeat with μ^{k+1} replacing μ^k .

- **Result:** For each k , we either have $\lambda^{k+1} < \lambda^k$ or we have policy improvement for the λ^k -SSP:

$$\lambda^{k+1} = \lambda^k, \quad h^{k+1}(i) \leq h^k(i), \quad i = 1, \dots, n.$$

The algorithm terminates with an optimal policy.

6.231 DYNAMIC PROGRAMMING

LECTURE 14

LECTURE OUTLINE

- Control of continuous-time Markov chains – Semi-Markov problems
- Problem formulation – Equivalence to discrete-time problems
- Discounted problems
- Average cost problems

CONTINUOUS-TIME MARKOV CHAINS

- Stationary system with finite number of states and controls
- State transitions occur at discrete times
- Control applied at these discrete times and stays constant between transitions
- Time between transitions is random
- Cost accumulates in continuous time (may also be incurred at the time of transition)
- **Example:** Admission control in a system with restricted capacity (e.g., a communication link)
 - Customer arrivals: a Poisson process
 - Customers entering the system, depart after exponentially distributed time
 - Upon arrival we must decide whether to admit or to block a customer
 - There is a cost for blocking a customer
 - For each customer that is in the system, there is a customer-dependent reward per unit time
 - Minimize time-discounted or average cost

PROBLEM FORMULATION

- $x(t)$ and $u(t)$: State and control at time t
- t_k : Time of k th transition ($t_0 = 0$)
- $x_k = x(t_k)$; $x(t) = x_k$ for $t_k \leq t < t_{k+1}$.
- $u_k = u(t_k)$; $u(t) = u_k$ for $t_k \leq t < t_{k+1}$.
- No transition probabilities; instead **transition distributions** (quantify the uncertainty about **both** transition time and next state)

$$Q_{ij}(\tau, u) = P\{t_{k+1} - t_k \leq \tau, x_{k+1} = j \mid x_k = i, u_k = u\}$$

- **Two important formulas:**

(1) Transition probabilities are specified by

$$p_{ij}(u) = P\{x_{k+1} = j \mid x_k = i, u_k = u\} = \lim_{\tau \rightarrow \infty} Q_{ij}(\tau, u)$$

(2) The Cumulative Distribution Function (CDF) of τ given i, j, u is (assuming $p_{ij}(u) > 0$)

$$P\{t_{k+1} - t_k \leq \tau \mid x_k = i, x_{k+1} = j, u_k = u\} = \frac{Q_{ij}(\tau, u)}{p_{ij}(u)}$$

Thus, $Q_{ij}(\tau, u)$ can be viewed as a “scaled CDF”

EXPONENTIAL TRANSITION DISTRIBUTIONS

- Important example of transition distributions:

$$Q_{ij}(\tau, u) = p_{ij}(u)(1 - e^{-\nu_i(u)\tau}),$$

where $p_{ij}(u)$ are transition probabilities, and $\nu_i(u)$ is called the *transition rate* at state i .

- **Interpretation:** If the system is in state i and control u is applied
 - the next state will be j with probability $p_{ij}(u)$
 - the time between the transition to state i and the transition to the next state j is exponentially distributed with parameter $\nu_i(u)$ (independently of j):

$$P\{\text{transition time interval} > \tau \mid i, u\} = e^{-\nu_i(u)\tau}$$

- The exponential distribution is **memoryless**. This implies that for a given policy, the system is a continuous-time Markov chain (the future depends on the past through the current state).
- Without the memoryless property, the Markov property holds only at the times of transition.

COST STRUCTURES

- There is cost $g(i, u)$ per unit time, i.e.

$$g(i, u)dt = \text{the cost incurred in time } dt$$

- There may be an extra “instantaneous” cost $\hat{g}(i, u)$ at the time of a transition (let’s ignore this for the moment)
- **Total discounted cost** of $\pi = \{\mu_0, \mu_1, \dots\}$ starting from state i (with discount factor $\beta > 0$)

$$\lim_{N \rightarrow \infty} E \left\{ \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} e^{-\beta t} g(x_k, \mu_k(x_k)) dt \mid x_0 = i \right\}$$

- **Average cost per unit time**

$$\lim_{N \rightarrow \infty} \frac{1}{E\{t_N\}} E \left\{ \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} g(x_k, \mu_k(x_k)) dt \mid x_0 = i \right\}$$

- We will see that **both problems have equivalent discrete-time versions.**

A NOTE ON NOTATION

- The scaled CDF $Q_{ij}(\tau, u)$ can be used to model discrete, continuous, and mixed distributions for the transition time τ .
- Generally, expected values of functions of τ can be written as integrals involving $dQ_{ij}(\tau, u)$. For example, the conditional expected value of τ given i, j , and u is written as

$$E\{\tau \mid i, j, u\} = \int_0^{\infty} \tau \frac{dQ_{ij}(\tau, u)}{p_{ij}(u)}$$

- If $Q_{ij}(\tau, u)$ is continuous with respect to τ , its derivative

$$q_{ij}(\tau, u) = \frac{dQ_{ij}}{d\tau}(\tau, u)$$

can be viewed as a “scaled” density function. Expected values of functions of τ can then be written in terms of $q_{ij}(\tau, u)$. For example

$$E\{\tau \mid i, j, u\} = \int_0^{\infty} \tau \frac{q_{ij}(\tau, u)}{p_{ij}(u)} d\tau$$

- If $Q_{ij}(\tau, u)$ is discontinuous and “staircase-like,” expected values can be written as summations.

DISCOUNTED CASE - COST CALCULATION

- For a policy $\pi = \{\mu_0, \mu_1, \dots\}$, write

$$J_\pi(i) = E\{\text{1st transition cost}\} + E\{e^{-\beta\tau} J_{\pi_1}(j) \mid i, \mu_0(i)\}$$

where $E\{\text{1st transition cost}\} = E\left\{\int_0^\tau e^{-\beta t} g(i, u) dt\right\}$
and $J_{\pi_1}(j)$ is the cost-to-go of $\pi_1 = \{\mu_1, \mu_2, \dots\}$

- We calculate the two costs in the RHS. The $E\{\text{1st transition cost}\}$, if u is applied at state i , is

$$\begin{aligned} G(i, u) &= E_j\left\{E_\tau\{\text{1st transition cost} \mid j\}\right\} \\ &= \sum_{j=1}^n p_{ij}(u) \int_0^\infty \left(\int_0^\tau e^{-\beta t} g(i, u) dt\right) \frac{dQ_{ij}(\tau, u)}{p_{ij}(u)} \\ &= g(i, u) \sum_{j=1}^n \int_0^\infty \frac{1 - e^{-\beta\tau}}{\beta} dQ_{ij}(\tau, u) \end{aligned}$$

- Thus the $E\{\text{1st transition cost}\}$ is

$$G(i, \mu_0(i)) = g(i, \mu_0(i)) \sum_{j=1}^n \int_0^\infty \frac{1 - e^{-\beta\tau}}{\beta} dQ_{ij}(\tau, \mu_0(i))$$

(The summation term can be viewed as a “discounted length of the transition interval $t_1 - t_0$ ”.)

COST CALCULATION (CONTINUED)

- Also the expected (discounted) cost from the next state j is

$$\begin{aligned}
 & E\{e^{-\beta\tau} J_{\pi_1}(j) \mid i, \mu_0(i)\} \\
 &= E_j\{E\{e^{-\beta\tau} \mid i, \mu_0(i), j\} J_{\pi_1}(j) \mid i, \mu_0(i)\} \\
 &= \sum_{j=1}^n p_{ij}(\mu_0(i)) \left(\int_0^\infty e^{-\beta\tau} \frac{dQ_{ij}(\tau, \mu_0(i))}{p_{ij}(\mu_0(i))} \right) J_{\pi_1}(j) \\
 &= \sum_{j=1}^n m_{ij}(\mu_0(i)) J_{\pi_1}(j)
 \end{aligned}$$

where $m_{ij}(u)$ is given by

$$m_{ij}(u) = \int_0^\infty e^{-\beta\tau} dQ_{ij}(\tau, u) \left(< \int_0^\infty dQ_{ij}(\tau, u) = p_{ij}(u) \right)$$

and can be viewed as the “effective discount factor” [the analog of $\alpha p_{ij}(u)$ in discrete-time case].

- So $J_\pi(i)$ can be written as

$$J_\pi(i) = G(i, \mu_0(i)) + \sum_{j=1}^n m_{ij}(\mu_0(i)) J_{\pi_1}(j)$$

i.e., the (continuous-time discounted) cost of 1st period, plus the (continuous-time discounted) cost-to-go from the next state.

EQUIVALENCE TO AN SSP

- Similar to the discrete-time case, introduce a stochastic shortest path problem with an artificial termination state t
- Under control u , from state i the system moves to state j with probability $m_{ij}(u)$ and to the termination state t with probability $1 - \sum_{j=1}^n m_{ij}(u)$
- **Bellman's equation:** For $i = 1, \dots, n$,

$$J^*(i) = \min_{u \in U(i)} \left[G(i, u) + \sum_{j=1}^n m_{ij}(u) J^*(j) \right]$$

- Analogs of value iteration, policy iteration, and linear programming.
- If in addition to the cost per unit time g , there is an extra (instantaneous) one-stage cost $\hat{g}(i, u)$, Bellman's equation becomes

$$J^*(i) = \min_{u \in U(i)} \left[\hat{g}(i, u) + G(i, u) + \sum_{j=1}^n m_{ij}(u) J^*(j) \right]$$

MANUFACTURER'S EXAMPLE REVISITED

- A manufacturer receives orders with interarrival times uniformly distributed in $[0, \tau_{\max}]$.
- He may process all unfilled orders at cost $K > 0$, or process none. The cost per unit time of an unfilled order is c . Max number of unfilled orders is n .
- The nonzero transition distributions are

$$Q_{i1}(\tau, \text{Fill}) = Q_{i(i+1)}(\tau, \text{Not Fill}) = \min \left[1, \frac{\tau}{\tau_{\max}} \right]$$

- The one-stage expected cost G is

$$G(i, \text{Fill}) = 0, \quad G(i, \text{Not Fill}) = \gamma c i,$$

where

$$\gamma = \sum_{j=1}^n \int_0^{\infty} \frac{1 - e^{-\beta\tau}}{\beta} dQ_{ij}(\tau, u) = \int_0^{\tau_{\max}} \frac{1 - e^{-\beta\tau}}{\beta\tau_{\max}} d\tau$$

- There is an “instantaneous” cost

$$\hat{g}(i, \text{Fill}) = K, \quad \hat{g}(i, \text{Not Fill}) = 0$$

MANUFACTURER'S EXAMPLE CONTINUED

- The “effective discount factors” $m_{ij}(u)$ in Bellman's Equation are

$$m_{i1}(\text{Fill}) = m_{i(i+1)}(\text{Not Fill}) = \alpha,$$

where

$$\alpha = \int_0^{\infty} e^{-\beta\tau} dQ_{ij}(\tau, u) = \int_0^{\tau_{\max}} \frac{e^{-\beta\tau}}{\tau_{\max}} d\tau = \frac{1 - e^{-\beta\tau_{\max}}}{\beta\tau_{\max}}$$

- Bellman's equation has the form

$$J^*(i) = \min[K + \alpha J^*(1), \gamma ci + \alpha J^*(i+1)], \quad i = 1, 2, \dots$$

- As in the discrete-time case, we can conclude that there exists an optimal threshold i^* :

fill the orders \iff their number i exceeds i^*

AVERAGE COST

- Minimize

$$\lim_{N \rightarrow \infty} \frac{1}{E\{t_N\}} E \left\{ \int_0^{t_N} g(x(t), u(t)) dt \right\}$$

assuming there is a special state that is “recurrent under all policies”

- Total expected cost of a transition

$$G(i, u) = g(i, u)\bar{\tau}_i(u),$$

where $\bar{\tau}_i(u)$: Expected transition time.

- We now apply the SSP argument used for the discrete-time case. Divide trajectory into cycles marked by successive visits to n . The cost at (i, u) is $G(i, u) - \lambda^*\bar{\tau}_i(u)$, where λ^* is the optimal expected cost per unit time. Each cycle is viewed as a state trajectory of a corresponding SSP problem with the termination state being essentially n .

- So Bellman’s Eq. for the average cost problem:

$$h^*(i) = \min_{u \in U(i)} \left[G(i, u) - \lambda^*\bar{\tau}_i(u) + \sum_{j=1}^n p_{ij}(u)h^*(j) \right]$$

MANUFACTURER EXAMPLE/AVERAGE COST

- The expected transition times are

$$\bar{\tau}_i(\text{Fill}) = \bar{\tau}_i(\text{Not Fill}) = \frac{\tau_{\max}}{2}$$

the expected transition cost is

$$G(i, \text{Fill}) = 0, \quad G(i, \text{Not Fill}) = \frac{ci\tau_{\max}}{2}$$

and there is also the “instantaneous” cost

$$\hat{g}(i, \text{Fill}) = K, \quad \hat{g}(i, \text{Not Fill}) = 0$$

- Bellman’s equation:

$$h^*(i) = \min \left[K - \lambda^* \frac{\tau_{\max}}{2} + h^*(1), \right. \\ \left. ci \frac{\tau_{\max}}{2} - \lambda^* \frac{\tau_{\max}}{2} + h^*(i+1) \right]$$

- Again it can be shown that a threshold policy is optimal.

6.231 DYNAMIC PROGRAMMING

LECTURE 15

LECTURE OUTLINE

- We start a nine-lecture sequence on advanced infinite horizon DP and approximation methods
- We allow infinite state space, so the stochastic shortest path framework cannot be used any more
- Results are rigorous assuming a countable disturbance space
 - This includes deterministic problems with arbitrary state space, and countable state Markov chains
 - Otherwise the mathematics of measure theory make analysis difficult, although the final results are essentially the same as for countable disturbance space
- We use **Volume II** starting with the discounted problem (Chapter 1)
- The central mathematical structure is that the DP mapping is a contraction mapping (instead of existence of a termination state)

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, \dots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \dots\}$

$$J_\pi(x_0) = \lim_{N \rightarrow \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}$$

with $\alpha < 1$, and for some M , we have $|g(x, u, w)| \leq M$ for all (x, u, w)

- **Shorthand notation for DP mappings** (operate on functions of state to produce other functions)

$$(TJ)(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + \alpha J(f(x, u, w)) \right\}, \quad \forall x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ .

- For any stationary policy μ

$$(T_\mu J)(x) = E_w \left\{ g(x, \mu(x), w) + \alpha J(f(x, \mu(x), w)) \right\}, \quad \forall x$$

“SHORTHAND” THEORY – A SUMMARY

- **Cost function expressions** [with $J_0(x) \equiv 0$]

$$J_\pi(x) = \lim_{k \rightarrow \infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0)(x), \quad J_\mu(x) = \lim_{k \rightarrow \infty} (T_\mu^k J_0)(x)$$

- **Bellman’s equation:** $J^* = T J^*$, $J_\mu = T_\mu J_\mu$
- **Optimality condition:**

$$\mu: \text{optimal} \quad \langle == \rangle \quad T_\mu J^* = T J^*$$

- **Value iteration:** For any (bounded) J and all x ,

$$J^*(x) = \lim_{k \rightarrow \infty} (T^k J)(x)$$

- **Policy iteration:** Given μ^k ,
 - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

- Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$$

TWO KEY PROPERTIES

- **Monotonicity property:** For any functions J and J' such that $J(x) \leq J'(x)$ for all x , and any μ

$$(TJ)(x) \leq (TJ')(x), \quad \forall x,$$

$$(T_\mu J)(x) \leq (T_\mu J')(x), \quad \forall x.$$

- **Constant Shift property:** For any J , any scalar r , and any μ

$$(T(J + re))(x) = (TJ)(x) + \alpha r, \quad \forall x,$$

$$(T_\mu(J + re))(x) = (T_\mu J)(x) + \alpha r, \quad \forall x,$$

where e is the unit function [$e(x) \equiv 1$].

- These properties hold for almost all DP models.
- A third important property that holds for some (but not all) DP models is that T and T_μ are **contraction mappings** (more on this later).

CONVERGENCE OF VALUE ITERATION

- If $J_0 \equiv 0$,

$$J^*(x) = \lim_{N \rightarrow \infty} (T^N J_0)(x), \quad \text{for all } x$$

Proof: For any initial state x_0 , and policy $\pi = \{\mu_0, \mu_1, \dots\}$,

$$\begin{aligned} J_\pi(x_0) &= E \left\{ \sum_{k=0}^{\infty} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\} \\ &= E \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\} \\ &\quad + E \left\{ \sum_{k=N}^{\infty} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\} \end{aligned}$$

The tail portion satisfies

$$\left| E \left\{ \sum_{k=N}^{\infty} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\} \right| \leq \frac{\alpha^N M}{1 - \alpha},$$

where $M \geq |g(x, u, w)|$. Take the min over π of both sides. **Q.E.D.**

BELLMAN'S EQUATION

- The optimal cost function J^* satisfies Bellman's Eq., i.e. $J^* = TJ^*$.

Proof: For all x and N ,

$$J^*(x) - \frac{\alpha^N M}{1 - \alpha} \leq (T^N J_0)(x) \leq J^*(x) + \frac{\alpha^N M}{1 - \alpha},$$

where $J_0(x) \equiv 0$ and $M \geq |g(x, u, w)|$. Applying T to this relation, and using Monotonicity and Constant Shift,

$$\begin{aligned} (TJ^*)(x) - \frac{\alpha^{N+1} M}{1 - \alpha} &\leq (T^{N+1} J_0)(x) \\ &\leq (TJ^*)(x) + \frac{\alpha^{N+1} M}{1 - \alpha} \end{aligned}$$

Taking the limit as $N \rightarrow \infty$ and using the fact

$$\lim_{N \rightarrow \infty} (T^{N+1} J_0)(x) = J^*(x)$$

we obtain $J^* = TJ^*$. **Q.E.D.**

THE CONTRACTION PROPERTY

- **Contraction property:** For any bounded functions J and J' , and any μ ,

$$\max_x |(TJ)(x) - (TJ')(x)| \leq \alpha \max_x |J(x) - J'(x)|,$$

$$\max_x |(T_\mu J)(x) - (T_\mu J')(x)| \leq \alpha \max_x |J(x) - J'(x)|.$$

Proof: Denote $c = \max_{x \in S} |J(x) - J'(x)|$. Then

$$J(x) - c \leq J'(x) \leq J(x) + c, \quad \forall x$$

Apply T to both sides, and use the Monotonicity and Constant Shift properties:

$$(TJ)(x) - \alpha c \leq (TJ')(x) \leq (TJ)(x) + \alpha c, \quad \forall x$$

Hence

$$|(TJ)(x) - (TJ')(x)| \leq \alpha c, \quad \forall x.$$

Q.E.D.

IMPLICATIONS OF CONTRACTION PROPERTY

- We can strengthen our earlier result:
- Bellman's equation $J = TJ$ has a unique solution, namely J^* , and for any bounded J , we have

$$\lim_{k \rightarrow \infty} (T^k J)(x) = J^*(x), \quad \forall x$$

Proof: Use

$$\begin{aligned} \max_x |(T^k J)(x) - J^*(x)| &= \max_x |(T^k J)(x) - (T^k J^*)(x)| \\ &\leq \alpha^k \max_x |J(x) - J^*(x)| \end{aligned}$$

- **Special Case:** For each stationary μ , J_μ is the unique solution of $J = T_\mu J$ and

$$\lim_{k \rightarrow \infty} (T_\mu^k J)(x) = J_\mu(x), \quad \forall x,$$

for any bounded J .

- **Convergence rate:** For all k ,

$$\max_x |(T^k J)(x) - J^*(x)| \leq \alpha^k \max_x |J(x) - J^*(x)|$$

NEC. AND SUFFICIENT OPT. CONDITION

- A stationary policy μ is optimal if and only if $\mu(x)$ attains the minimum in Bellman's equation for each x ; i.e.,

$$TJ^* = T_\mu J^*.$$

Proof: If $TJ^* = T_\mu J^*$, then using Bellman's equation ($J^* = TJ^*$), we have

$$J^* = T_\mu J^*,$$

so by uniqueness of the fixed point of T_μ , we obtain $J^* = J_\mu$; i.e., μ is optimal.

- Conversely, if the stationary policy μ is optimal, we have $J^* = J_\mu$, so

$$J^* = T_\mu J^*.$$

Combining this with Bellman's equation ($J^* = TJ^*$), we obtain $TJ^* = T_\mu J^*$. **Q.E.D.**

COMPUTATIONAL METHODS - AN OVERVIEW

- Typically must work with a **finite-state system**. Possibly an approximation of the original system.
- **Value iteration** and variants
 - Gauss-Seidel and asynchronous versions
- **Policy iteration** and variants
 - Combination with (possibly asynchronous) value iteration
 - “Optimistic” policy iteration
- **Linear programming**

$$\text{maximize } \sum_{i=1}^n J(i)$$

$$\text{subject to } J(i) \leq g(i, u) + \alpha \sum_{j=1}^n p_{ij}(u) J(j), \quad \forall (i, u)$$

- **Versions with subspace approximation:** use in place of $J(i)$ a low-dim. basis function representation, with state features $\phi_m(i)$, $m = 1, \dots, s$

$$\tilde{J}(i, r) = \sum_{m=1}^s r_m \phi_m(i)$$

and modify the basic methods appropriately.

USING Q-FACTORS I

- Let the states be $i = 1, \dots, n$. We can write Bellman's equation as

$$J^*(i) = \min_{u \in U(i)} Q^*(i, u) \quad i = 1, \dots, n,$$

where

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

for all (i, u)

- $Q^*(i, u)$ is called the **optimal Q-factor** of (i, u)
- Q-factors have optimal cost interpretation in an “augmented” problem whose states are i **and** $(i, u), u \in U(i)$ - the optimal cost vector is (J^*, Q^*)
- The Bellman Eq. is $J^* = TJ^*, Q^* = FQ^*$ where

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

- It has a unique solution.

USING Q-FACTORS II

- We can equivalently write the VI method as

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i, u), \quad i = 1, \dots, n,$$

where Q_{k+1} is generated for all i and $u \in U(i)$ by

$$Q_{k+1}(i, u) = \sum_{j=1}^n p_{ij}(u) \left(g(i, u, j) + \alpha \min_{v \in U(j)} Q_k(j, v) \right)$$

or $J_{k+1} = TJ_k$, $Q_{k+1} = FQ_k$.

- It requires equal amount of computation ... it just needs more storage
- Having optimal Q-factors is convenient when implementing an optimal policy on-line by

$$\mu^*(i) = \min_{u \in U(i)} Q^*(i, u)$$

- Once $Q^*(i, u)$ are known, the model [g and $p_{ij}(u)$] is not needed. **Model-free operation**
- Later we will see how stochastic/sampling methods can be used to calculate (approximations of) $Q^*(i, u)$ using a simulator of the system (**no model needed**)

6.231 DYNAMIC PROGRAMMING

LECTURE 16

LECTURE OUTLINE

- Review of basic theory of discounted problems
- Monotonicity and contraction properties
- Contraction mappings in DP
- Discounted problems: Countable state space with unbounded costs
- Generalized discounted DP

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, \dots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \dots\}$

$$J_\pi(x_0) = \lim_{N \rightarrow \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}$$

with $\alpha < 1$, and for some M , we have $|g(x, u, w)| \leq M$ for all (x, u, w)

- **Shorthand notation for DP mappings** (operate on functions of state to produce other functions)

$$(TJ)(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + \alpha J(f(x, u, w)) \right\}, \quad \forall x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ .

- For any stationary policy μ

$$(T_\mu J)(x) = E_w \left\{ g(x, \mu(x), w) + \alpha J(f(x, \mu(x), w)) \right\}, \quad \forall x$$

“SHORTHAND” THEORY – A SUMMARY

- **Cost function expressions** [with $J_0(x) \equiv 0$]

$$J_\pi(x) = \lim_{k \rightarrow \infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0)(x), \quad J_\mu(x) = \lim_{k \rightarrow \infty} (T_\mu^k J_0)(x)$$

- **Bellman’s equation:** $J^* = T J^*$, $J_\mu = T_\mu J_\mu$
- **Optimality condition:**

$$\mu: \text{optimal} \quad \Leftrightarrow \quad T_\mu J^* = T J^*$$

- **Value iteration:** For any (bounded) J and all x ,

$$J^*(x) = \lim_{k \rightarrow \infty} (T^k J)(x)$$

- **Policy iteration:** Given μ^k ,
 - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

- Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$$

MAJOR PROPERTIES

- **Monotonicity property:** For any functions J and J' on the state space X such that $J(x) \leq J'(x)$ for all $x \in X$, and any μ

$$(TJ)(x) \leq (TJ')(x), \quad \forall x \in X,$$

$$(T_\mu J)(x) \leq (T_\mu J')(x), \quad \forall x \in X.$$

- **Contraction property:** For any bounded functions J and J' , and any μ ,

$$\max_x |(TJ)(x) - (TJ')(x)| \leq \alpha \max_x |J(x) - J'(x)|,$$

$$\max_x |(T_\mu J)(x) - (T_\mu J')(x)| \leq \alpha \max_x |J(x) - J'(x)|.$$

- The contraction property can be written in shorthand as

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

where for any bounded function J , we denote by $\|J\|$ the sup-norm

$$\|J\| = \max_{x \in X} |J(x)|.$$

CONTRACTION MAPPINGS

- Given a real vector space Y with a norm $\|\cdot\|$ (see text for definitions).
- A function $F : Y \mapsto Y$ is said to be a *contraction mapping* if for some $\rho \in (0, 1)$, we have

$$\|Fy - Fz\| \leq \rho\|y - z\|, \quad \text{for all } y, z \in Y.$$

ρ is called the *modulus of contraction* of F .

- **Linear case, $Y = \Re^n$:** $Fy = Ay + b$ is a contraction if and only if all eigenvalues of A are strictly within the unit circle.
- For $m > 1$, we say that F is an *m -stage contraction* if F^m is a contraction.
- **Important example:** Let X be a set (e.g., state space in DP), $v : X \mapsto \Re$ be a positive-valued function. Let $B(X)$ be the set of all functions $J : X \mapsto \Re$ such that $J(s)/v(s)$ is bounded over s .
- The *weighted sup-norm* on $B(X)$:

$$\|J\| = \max_{s \in X} \frac{|J(s)|}{v(s)}.$$

- **Important special case:** The discounted problem mappings T and T_μ [for $v(s) \equiv 1$, $\rho = \alpha$].

A DP-LIKE CONTRACTION MAPPING

- Let $X = \{1, 2, \dots\}$, and let $F : B(X) \mapsto B(X)$ be a linear mapping of the form

$$(FJ)(i) = b(i) + \sum_{j \in X} a(i, j) J(j), \quad \forall i$$

where $b(i)$ and $a(i, j)$ are some scalars. Then F is a contraction with modulus ρ if

$$\frac{\sum_{j \in X} |a(i, j)| v(j)}{v(i)} \leq \rho, \quad \forall i$$

[Think of the special case where $a(i, j)$ are the transition probs. of a policy].

- Let $F : B(X) \mapsto B(X)$ be a mapping of the form

$$(FJ)(i) = \min_{\mu \in M} (F_{\mu} J)(i), \quad \forall i$$

where M is parameter set, and for each $\mu \in M$, F_{μ} is a contraction mapping from $B(X)$ to $B(X)$ with modulus ρ . Then F is a contraction mapping with modulus ρ .

CONTRACTION MAPPING FIXED-POINT TH.

- **Contraction Mapping Fixed-Point Theorem:** If $F : B(X) \mapsto B(X)$ is a contraction with modulus $\rho \in (0, 1)$, then there exists a unique $J^* \in B(X)$ such that

$$J^* = FJ^*.$$

Furthermore, if J is any function in $B(X)$, then $\{F^k J\}$ converges to J^* and we have

$$\|F^k J - J^*\| \leq \rho^k \|J - J^*\|, \quad k = 1, 2, \dots$$

- Similar result if F is an m -stage contraction mapping.
- This is a special case of a general result for contraction mappings $F : Y \mapsto Y$ over normed vector spaces Y that are *complete*: every sequence $\{y_k\}$ that is Cauchy (satisfies $\|y_m - y_n\| \rightarrow 0$ as $m, n \rightarrow \infty$) converges.
- The space $B(X)$ is complete [see the text (Section 1.5) for a proof].

GENERAL FORMS OF DISCOUNTED DP

- **Monotonicity assumption:** If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x, u, J) \leq H(x, u, J'), \quad \forall x \in X, u \in U(x)$$

- **Contraction assumption:**
 - For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
 - For some $\alpha \in (0, 1)$ and all $J, J' \in B(X)$, T_μ satisfies

$$\|T_\mu J - T_\mu J'\| \leq \alpha \|J - J'\|$$

- We can show all the standard analytical and computational results of discounted DP based on these two assumptions.
- With just the monotonicity assumption (as in shortest path problem) we can still show various forms of the basic results under appropriate assumptions (like in the SSP problem).

EXAMPLES

- Discounted problems

$$H(x, u, J) = E \{ g(x, u, w) + \alpha J(f(x, u, w)) \}$$

- Discounted Semi-Markov Problems

$$H(x, u, J) = G(x, u) + \sum_{y=1}^n m_{xy}(u) J(y)$$

where m_{xy} are “discounted” transition probabilities, defined by the transition distributions

- Shortest Path Problems

$$H(x, u, J) = \begin{cases} a_{xu} + J(u) & \text{if } u \neq d, \\ a_{xd} & \text{if } u = d \end{cases}$$

where d is the destination

- Minimax Problems

$$H(x, u, J) = \max_{w \in W(x, u)} [g(x, u, w) + \alpha J(f(x, u, w))]$$

GENERAL FORMS OF DISCOUNTED DP

- **Monotonicity assumption:** If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x, u, J) \leq H(x, u, J'), \quad \forall x \in X, u \in U(x)$$

- **Contraction assumption:**
 - For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
 - For some $\alpha \in (0, 1)$ and all $J, J' \in B(X)$, H satisfies

$$|H(x, u, J) - H(x, u, J')| \leq \alpha \max_{y \in X} |J(y) - J'(y)|$$

for all $x \in X$ and $u \in U(x)$.

- We can show all the standard analytical and computational results of discounted DP based on these two assumptions
- With just the monotonicity assumption (as in shortest path problem) we can still show various forms of the basic results under appropriate assumptions (like in the SSP problem)

RESULTS USING CONTRACTION

- The mappings T_μ and T are sup-norm contraction mappings with modulus α over $B(X)$, and have unique fixed points in $B(X)$, denoted J_μ and J^* , respectively (cf. **Bellman's equation**). *Proof:* From contraction assumption and fixed point th.
- For any $J \in B(X)$ and $\mu \in \mathcal{M}$,

$$\lim_{k \rightarrow \infty} T_\mu^k J = J_\mu, \quad \lim_{k \rightarrow \infty} T^k J = J^*$$

(cf. **convergence of value iteration**). *Proof:* From contraction property of T_μ and T .

- We have $T_\mu J^* = T J^*$ if and only if $J_\mu = J^*$ (cf. **optimality condition**). *Proof:* $T_\mu J^* = T J^*$, then $T_\mu J^* = J^*$, implying $J^* = J_\mu$. Conversely, if $J_\mu = J^*$, then $T_\mu J^* = T_\mu J_\mu = J_\mu = J^* = T J^*$.
- **Useful bound for J_μ :** For all $J \in B(X)$, $\mu \in \mathcal{M}$

$$\|J_\mu - J\| \leq \frac{\|T_\mu J - J\|}{1 - \alpha}$$

Proof:

$$\|T_\mu^k J - J\| \leq \sum_{\ell=1}^k \|T_\mu^\ell J - T_\mu^{\ell-1} J\| \leq \|T_\mu J - J\| \sum_{\ell=1}^k \alpha^{\ell-1}$$

RESULTS USING MON. AND CONTRACTION

- **Optimality of fixed point:**

$$J^*(x) = \min_{\mu \in \mathcal{M}} J_\mu(x), \quad \forall x \in X$$

- Furthermore, for every $\epsilon > 0$, there exists $\mu_\epsilon \in \mathcal{M}$ such that

$$J^*(x) \leq J_{\mu_\epsilon}(x) \leq J^*(x) + \epsilon, \quad \forall x \in X$$

- **Nonstationary policies:** Consider the set Π of all sequences $\pi = \{\mu_0, \mu_1, \dots\}$ with $\mu_k \in \mathcal{M}$ for all k , and define

$$J_\pi(x) = \liminf_{k \rightarrow \infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J)(x), \quad \forall x \in X,$$

with J being any function (the choice of J does not matter)

- We have

$$J^*(x) = \min_{\pi \in \Pi} J_\pi(x), \quad \forall x \in X$$

6.231 DYNAMIC PROGRAMMING

LECTURE 17

LECTURE OUTLINE

- Review of computational theory of discounted problems
- Value iteration (VI), policy iteration (PI)
- Optimistic PI
- Computational methods for generalized discounted DP
- Asynchronous algorithms

DISCOUNTED PROBLEMS

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, \dots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \dots\}$

$$J_\pi(x_0) = \lim_{N \rightarrow \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}$$

- **Shorthand notation for DP mappings** (n -state Markov chain case)

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

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ .

- For any stationary policy μ

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

Note: **T_μ is linear** [in short $T_\mu J = P_\mu(g_\mu + \alpha J)$].

“SHORTHAND” THEORY – A SUMMARY

- **Cost function expressions** (with $J_0 \equiv 0$)

$$J_\pi = \lim_{k \rightarrow \infty} T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0, \quad J_\mu = \lim_{k \rightarrow \infty} T_\mu^k J_0$$

- **Bellman’s equation:** $J^* = T J^*$, $J_\mu = T_\mu J_\mu$

- **Optimality condition:**

$$\mu: \text{optimal} \quad \langle == \rangle \quad T_\mu J^* = T J^*$$

- **Contraction:** $\|T J_1 - T J_2\| \leq \alpha \|J_1 - J_2\|$

- **Value iteration:** For any (bounded) J

$$J^* = \lim_{k \rightarrow \infty} T^k J$$

- **Policy iteration:** Given μ^k ,

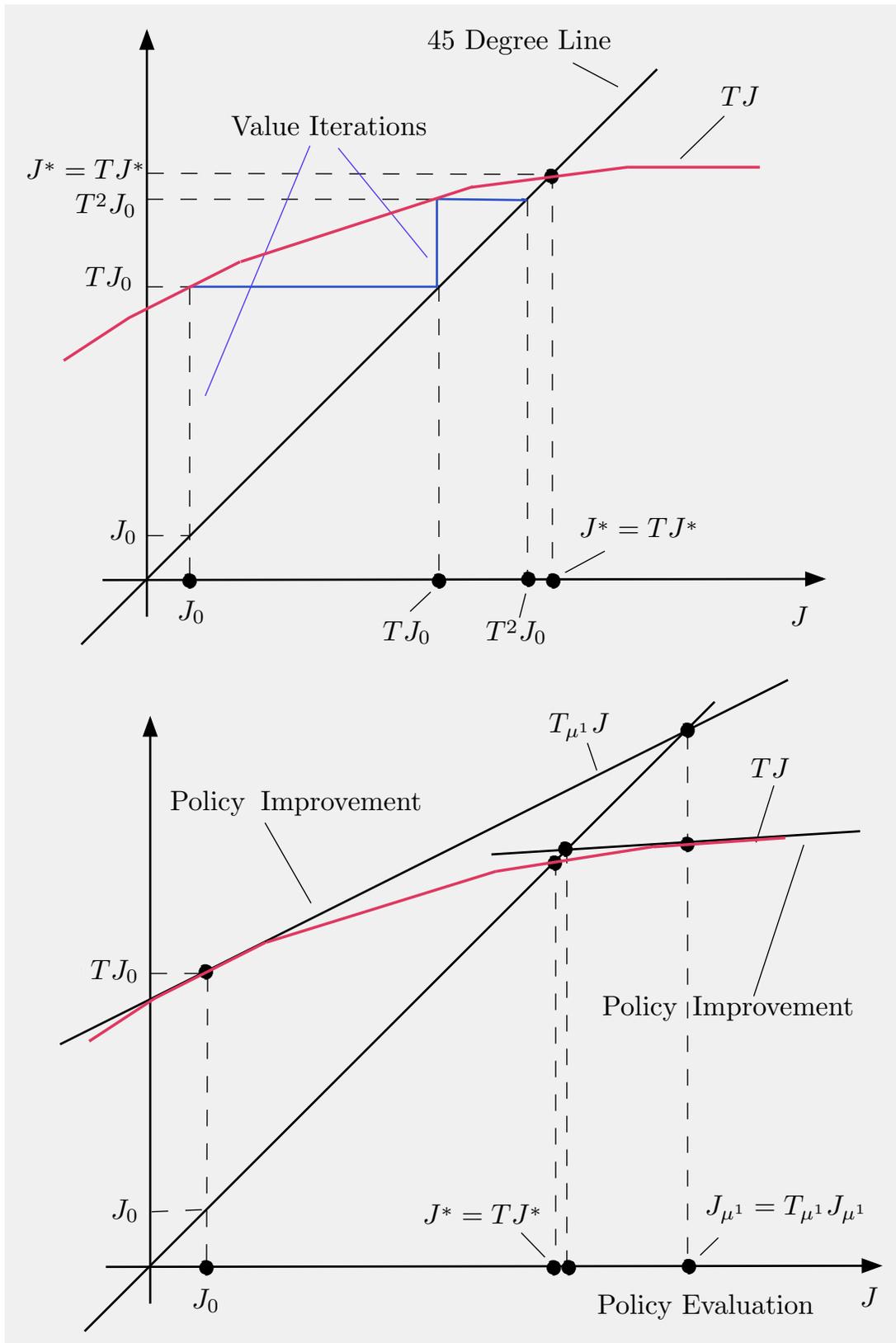
- **Policy evaluation:** Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

- **Policy improvement:** Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$$

INTERPRETATION OF VI AND PI



VI AND PI METHODS FOR Q-LEARNING

- We can write Bellman's equation as

$$J^*(i) = \min_{u \in U(i)} Q^*(i, u) \quad i = 1, \dots, n,$$

where Q^* is the vector of **optimal Q-factors** the unique solution of $Q = FQ$ where

$$(FQ)(i, u) = \sum_{j=1}^n p_{ij}(u) \left(g(i, u, j) + \alpha \min_{v \in U(j)} Q(j, v) \right)$$

- VI and PI for Q-factors are mathematically equivalent to VI and PI for costs.
- They require equal amount of computation ... they just need more storage.
- For example, we can write the VI method as

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i, u), \quad i = 1, \dots, n,$$

where Q_{k+1} is generated for all i and $u \in U(i)$ by

$$Q_{k+1}(i, u) = \sum_{j=1}^n p_{ij}(u) \left(g(i, u, j) + \alpha \min_{v \in U(j)} Q_k(j, v) \right)$$

APPROXIMATE PI

- Suppose that the policy evaluation is approximate, according to,

$$\max_x |J_k(x) - J_{\mu^k}(x)| \leq \delta, \quad k = 0, 1, \dots$$

and policy improvement is approximate, according to,

$$\max_x |(T_{\mu^{k+1}} J_k)(x) - (T J_k)(x)| \leq \epsilon, \quad k = 0, 1, \dots$$

where δ and ϵ are some positive scalars.

- **Error Bound:** The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \rightarrow \infty} \max_{x \in \mathcal{S}} (J_{\mu^k}(x) - J^*(x)) \leq \frac{\epsilon + 2\alpha\delta}{(1 - \alpha)^2}$$

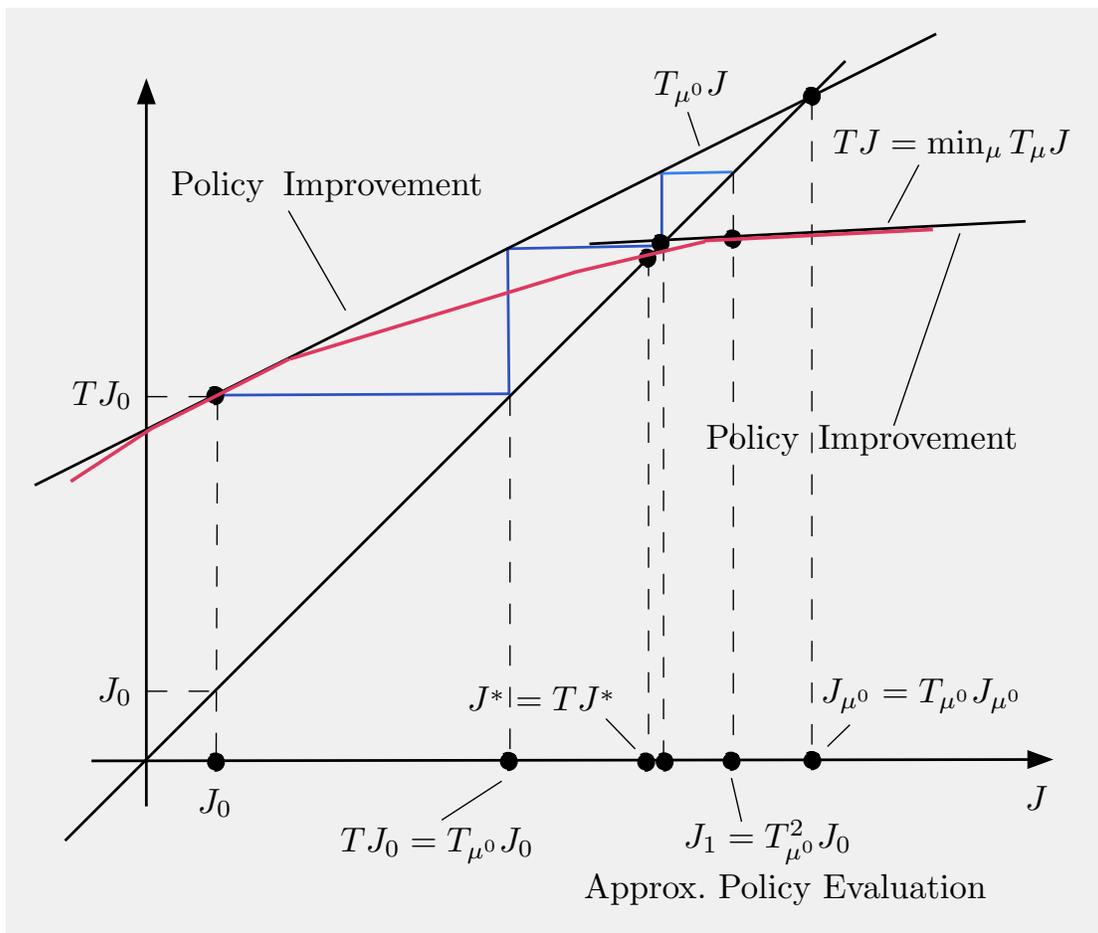
- **Typical practical behavior:** The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^* .

OPTIMISTIC PI

- This is PI, where policy evaluation is carried out by a finite number of VI
- **Shorthand definition:** For some integers m_k

$$T_{\mu^k} J_k = T J_k, \quad J_{k+1} = T_{\mu^k}^{m_k} J_k, \quad k = 0, 1, \dots$$

- If $m_k \equiv 1$ it becomes VI
- If $m_k = \infty$ it becomes PI
- For intermediate values of m_k , it is generally more efficient than either VI or PI



EXTENSIONS TO GENERALIZED DISC. DP

- All the preceding VI and PI methods extend to generalized discounted DP.
- **Summary:** For a mapping $H : X \times U \times R(X) \mapsto \mathfrak{R}$, consider

$$(TJ)(x) = \min_{u \in U(x)} H(x, u, J), \quad \forall x \in X.$$

$$(T_\mu J)(x) = H(x, \mu(x), J), \quad \forall x \in X.$$

- We want to find J^* such that

$$J^*(x) = \min_{u \in U(x)} H(x, u, J^*), \quad \forall x \in X$$

and a μ^* such that $T_{\mu^*} J^* = TJ^*$.

- Discounted, Discounted Semi-Markov, Minimax

$$H(x, u, J) = E \{ g(x, u, w) + \alpha J(f(x, u, w)) \}$$

$$H(x, u, J) = G(x, u) + \sum_{y=1}^n m_{xy}(u) J(y)$$

$$H(x, u, J) = \max_{w \in W(x, u)} [g(x, u, w) + \alpha J(f(x, u, w))]$$

ASSUMPTIONS AND RESULTS

- **Monotonicity assumption:** If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x, u, J) \leq H(x, u, J'), \quad \forall x \in X, u \in U(x)$$

- **Contraction assumption:**

- For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
- For some $\alpha \in (0, 1)$ and all $J, J' \in B(X)$, H satisfies

$$|H(x, u, J) - H(x, u, J')| \leq \alpha \max_{y \in X} |J(y) - J'(y)|$$

for all $x \in X$ and $u \in U(x)$.

- **Standard algorithmic results extend:**

- Generalized VI converges to J^* , the unique fixed point of T
- Generalized PI and optimistic PI generate $\{\mu^k\}$ such that

$$\lim_{k \rightarrow \infty} \|J_k - J^*\| = 0$$

ASYNCHRONOUS ALGORITHMS

- Motivation for asynchronous algorithms
 - Faster convergence
 - Parallel and distributed computation
 - Simulation-based implementations
- **General framework:** Partition X into disjoint nonempty subsets X_1, \dots, X_m , and use separate processor ℓ updating $J(x)$ for $x \in X_\ell$
- J be partitioned as

$$J = (J_1, \dots, J_m),$$

where J_ℓ is the restriction of J on the set X_ℓ .

- **Synchronous algorithm:**

$$J_\ell^{t+1}(x) = T(J_1^t, \dots, J_m^t)(x), \quad x \in X_\ell, \ell = 1, \dots, m$$

- **Asynchronous algorithm:** For some subsets of times \mathcal{R}_ℓ ,

$$J_\ell^{t+1}(x) = \begin{cases} T(J_1^{\tau_{\ell 1}(t)}, \dots, J_m^{\tau_{\ell m}(t)})(x) & \text{if } t \in \mathcal{R}_\ell, \\ J_\ell^t(x) & \text{if } t \notin \mathcal{R}_\ell \end{cases}$$

where $t - \tau_{\ell j}(t)$ are communication “delays”

ONE-STATE-AT-A-TIME ITERATIONS

- **Important special case:** Assume n “states”, a separate processor for each state, and no delays
- Generate a sequence of states $\{x^0, x^1, \dots\}$, generated in some way, possibly by simulation (each state is generated infinitely often)
- **Asynchronous VI:**

$$J_\ell^{t+1} = \begin{cases} T(J_1^t, \dots, J_n^t)(\ell) & \text{if } \ell = x^t, \\ J_\ell^t & \text{if } \ell \neq x^t, \end{cases}$$

where $T(J_1^t, \dots, J_n^t)(\ell)$ denotes the ℓ -th component of the vector

$$T(J_1^t, \dots, J_n^t) = T J^t,$$

and for simplicity we write J_ℓ^t instead of $J_\ell^t(\ell)$

- The special case where

$$\{x^0, x^1, \dots\} = \{1, \dots, n, 1, \dots, n, 1, \dots\}$$

is the **Gauss-Seidel method**

- We can show that $J^t \rightarrow J^*$ under the contraction assumption

6.231 DYNAMIC PROGRAMMING

LECTURE 18

LECTURE OUTLINE

- Analysis of asynchronous VI and PI for generalized discounted DP
- Undiscounted problems
- Stochastic shortest path problems (SSP)
- Proper and improper policies
- Analysis and computational methods for SSP
- Pathologies of SSP

REVIEW OF ASYNCHRONOUS ALGORITHMS

- **General framework:** Partition X into disjoint nonempty subsets X_1, \dots, X_m , and use separate processor ℓ updating $J(x)$ for $x \in X_\ell$
- J be partitioned as

$$J = (J_1, \dots, J_m),$$

where J_ℓ is the restriction of J on the set X_ℓ .

- **Synchronous algorithm:**

$$J_\ell^{t+1}(x) = T(J_1^t, \dots, J_m^t)(x), \quad x \in X_\ell, \ell = 1, \dots, m$$

- **Asynchronous algorithm:** For some subsets of times \mathcal{R}_ℓ ,

$$J_\ell^{t+1}(x) = \begin{cases} T(J_1^{\tau_{\ell 1}(t)}, \dots, J_m^{\tau_{\ell m}(t)})(x) & \text{if } t \in \mathcal{R}_\ell, \\ J_\ell^t(x) & \text{if } t \notin \mathcal{R}_\ell \end{cases}$$

where $t - \tau_{\ell j}(t)$ are communication “delays”

ASYNCHRONOUS CONV. THEOREM I

- Assume that for all $\ell, j = 1, \dots, m$, \mathcal{R}_ℓ is infinite and $\lim_{t \rightarrow \infty} \tau_{\ell j}(t) = \infty$
- **Proposition:** Let T have a unique fixed point J^* , and assume that there is a sequence of nonempty subsets $\{S(k)\} \subset R(X)$ with $S(k+1) \subset S(k)$ for all k , and with the following properties:

- (1) **Synchronous Convergence Condition:** Every sequence $\{J^k\}$ with $J^k \in S(k)$ for each k , converges pointwise to J^* . Moreover, we have

$$TJ \in S(k+1), \quad \forall J \in S(k), \quad k = 0, 1, \dots$$

- (2) **Box Condition:** For all k , $S(k)$ is a Cartesian product of the form

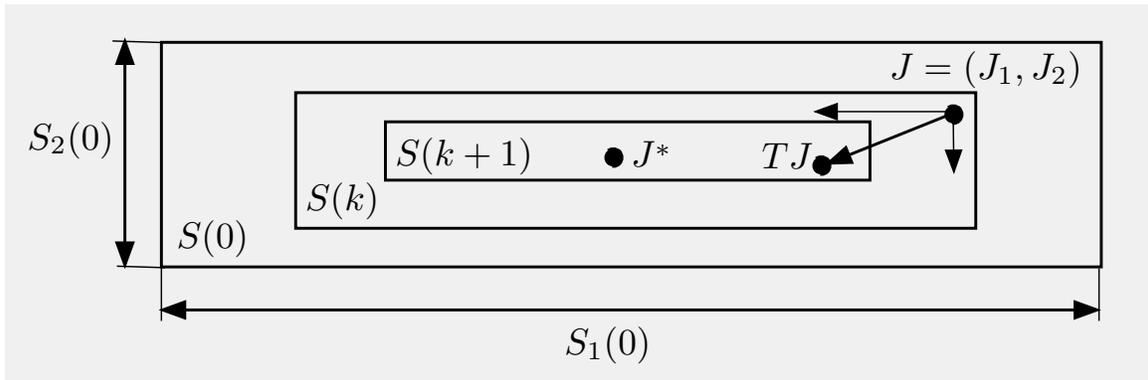
$$S(k) = S_1(k) \times \cdots \times S_m(k),$$

where $S_\ell(k)$ is a set of real-valued functions on X_ℓ , $\ell = 1, \dots, m$.

Then for every $J \in S(0)$, the sequence $\{J^t\}$ generated by the asynchronous algorithm converges pointwise to J^* .

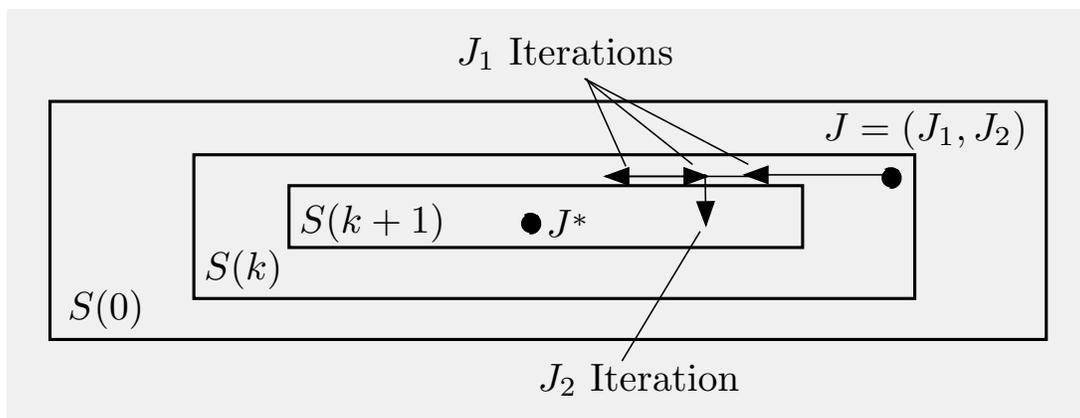
ASYNCHRONOUS CONV. THEOREM II

- Interpretation of assumptions:



A synchronous iteration from any J in $S(k)$ moves into $S(k + 1)$ (component-by-component)

- Convergence mechanism:



Key: “Independent” component-wise improvement. An asynchronous component iteration from any J in $S(k)$ moves into the corresponding component portion of $S(k + 1)$

UNDISCOUNTED PROBLEMS

- System: $x_{k+1} = f(x_k, u_k, w_k)$
- Cost of a policy $\pi = \{\mu_0, \mu_1, \dots\}$

$$J_\pi(x_0) = \lim_{N \rightarrow \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), w_k) \right\}$$

- Shorthand notation for DP mappings

$$(TJ)(x) = \min_{u \in U(x)} E_w \left\{ g(x, u, w) + J(f(x, u, w)) \right\}, \quad \forall x$$

- For any stationary policy μ

$$(T_\mu J)(x) = E_w \left\{ g(x, \mu(x), w) + J(f(x, \mu(x), w)) \right\}, \quad \forall x$$

- T and T_μ **need not be contractions in general**, but their monotonicity is helpful (see Ch. 4, Vol. II of text for an analysis).

- SSP problems provide a “soft boundary” between the easy finite-state discounted problems and the hard undiscounted problems.

- They share features of both.
- Some of the nice theory is recovered because of the termination state.

SSP THEORY SUMMARY I

- As earlier, we have a cost-free term. state t , a finite number of states $1, \dots, n$, and finite number of controls, but we will make weaker assumptions.
- Mappings T and T_μ (modified to account for termination state t):

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

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

- **Definition:** A stationary policy μ is called **proper**, if under μ , from every state i , there is a positive probability path that leads to t .
- **Important fact:** If μ is proper, T_μ is contraction with respect to some weighted max norm

$$\max_i \frac{1}{v_i} |(T_\mu J)(i) - (T_\mu J')(i)| \leq \rho_\mu \max_i \frac{1}{v_i} |J(i) - J'(i)|$$

- T is similarly a contraction if all μ are proper (the case discussed in the text, Ch. 7, Vol. I).

SSP THEORY SUMMARY II

- The theory can be pushed one step further. Assume that:

(a) There exists at least one proper policy

(b) For each improper μ , $J_\mu(i) = \infty$ for some i

- Then T is not necessarily a contraction, but:
 - J^* is the unique solution of Bellman's Equ.
 - μ^* is optimal if and only if $T_{\mu^*} J^* = T J^*$
 - $\lim_{k \rightarrow \infty} (T^k J)(i) = J^*(i)$ for all i
 - Policy iteration terminates with an optimal policy, if started with a proper policy
- **Example:** Deterministic shortest path problem with a single destination t .
 - States \Leftrightarrow nodes; Controls \Leftrightarrow arcs
 - Termination state \Leftrightarrow the destination
 - Assumption (a) \Leftrightarrow every node is connected to the destination
 - Assumption (b) \Leftrightarrow all cycle costs > 0

SSP ANALYSIS I

- For a proper policy μ , J_μ is the unique fixed point of T_μ , and $T_\mu^k J \rightarrow J_\mu$ for all J (holds by the theory of Vol. I, Section 7.2)
- **Key Fact:** A μ satisfying $J \geq T_\mu J$ for some $J \in \mathfrak{R}^n$ must be proper - true because

$$J \geq T_\mu^k J = P_\mu^k J + \sum_{m=0}^{k-1} P_\mu^m g_\mu$$

and some component of the term on the right blows up if μ is improper (by our assumptions).

- **Consequence:** T can have at most one fixed point within \mathfrak{R}^n .

Proof: If J and J' are two fixed points, select μ and μ' such that $J = TJ = T_\mu J$ and $J' = TJ' = T_{\mu'} J'$. By preceding assertion, μ and μ' must be proper, and $J = J_\mu$ and $J' = J_{\mu'}$. Also

$$J = T^k J \leq T_{\mu'}^k J \rightarrow J_{\mu'} = J'$$

Similarly, $J' \leq J$, so $J = J'$.

SSP ANALYSIS II

- We now **show that T has a fixed point**, and also that policy iteration converges.
- **Generate a sequence of proper policies $\{\mu_k\}$ by policy iteration** starting from a proper policy μ_0 .
- μ_1 is proper and $J_{\mu_0} \geq J_{\mu_1}$ since

$$J_{\mu_0} = T_{\mu_0} J_{\mu_0} \geq T J_{\mu_0} = T_{\mu_1} J_{\mu_0} \geq T_{\mu_1}^k J_{\mu_0} \geq J_{\mu_1}$$

- Thus $\{J_{\mu_k}\}$ is nonincreasing, some policy μ is repeated, with $J_\mu = T J_\mu$. So J_μ is a fixed point of T .
- Next **show $T^k J \rightarrow J_\mu$ for all J** , i.e., value iteration converges to the same limit as policy iteration. (Sketch: True if $J = J_\mu$, argue using the properness of μ to show that the terminal cost difference $J - J_\mu$ does not matter.)
- To **show $J_\mu = J^*$** , for any $\pi = \{\mu_0, \mu_1, \dots\}$

$$T_{\mu_0} \cdots T_{\mu_{k-1}} J_0 \geq T^k J_0,$$

where $J_0 \equiv 0$. Take \limsup as $k \rightarrow \infty$, to obtain $J_\pi \geq J_\mu$, so μ is optimal and $J_\mu = J^*$.

SSP ANALYSIS III

- **Contraction Property:** If all policies are proper (cf. Section 7.1, Vol. I), T_μ and T are contractions with respect to a weighted sup norm.

Proof: Consider a new SSP problem where the transition probabilities are the same as in the original, but the transition costs are all equal to -1 . Let \hat{J} be the corresponding optimal cost vector. For all μ ,

$$\hat{J}(i) = -1 + \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u) \hat{J}(j) \leq -1 + \sum_{j=1}^n p_{ij}(\mu(i)) \hat{J}(j)$$

For $v_i = -\hat{J}(i)$, we have $v_i \geq 1$, and for all μ ,

$$\sum_{j=1}^n p_{ij}(\mu(i)) v_j \leq v_i - 1 \leq \rho v_i, \quad i = 1, \dots, n,$$

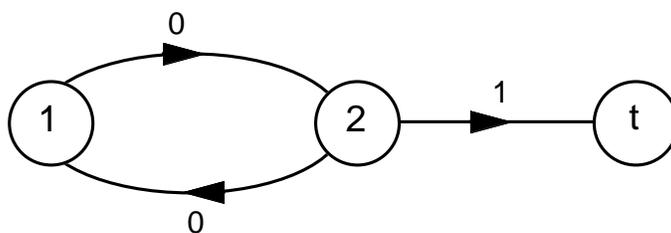
where

$$\rho = \max_{i=1, \dots, n} \frac{v_i - 1}{v_i} < 1.$$

This implies T_μ and T are contractions of modulus ρ for norm $\|J\| = \max_{i=1, \dots, n} |J(i)|/v_i$ (by the results of earlier lectures).

PATHOLOGIES I: DETERM. SHORTEST PATHS

- If there is a cycle with cost = 0, Bellman's equation has an **infinite number of solutions**. Example:



- We have $J^*(1) = J^*(2) = 0$.
- Bellman's equation is

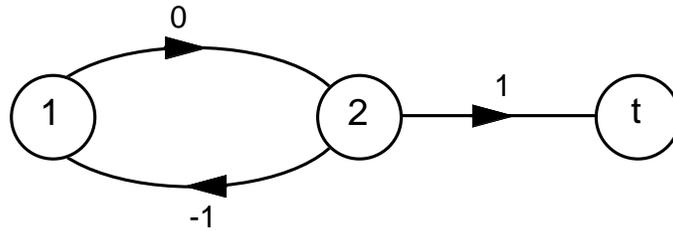
$$J(1) = J(2), \quad J(2) = \min[J(1), 1].$$

- It has J^* as solution.
- Set of solutions of Bellman's equation:

$$\{J \mid J(1) = J(2) \leq 1\}.$$

PATHOLOGIES II: DETERM. SHORTEST PATHS

- If there is a cycle with cost < 0 , **Bellman's equation has no solution** [among functions J with $-\infty < J(i) < \infty$ for all i]. Example:



- We have $J^*(1) = J^*(2) = -\infty$.
- Bellman's equation is

$$J(1) = J(2), \quad J(2) = \min[-1 + J(1), 1].$$

- There is no solution [among functions J with $-\infty < J(i) < \infty$ for all i].
- Bellman's equation has as solution $J^*(1) = J^*(2) = -\infty$ [within the larger class of functions $J(\cdot)$ that can take the value $-\infty$ for some (or all) states]. This situation can be generalized (see Chapter 4 of Vol. 2 of the text).

PATHOLOGIES III: THE BLACKMAILER

- Two states, state 1 and the termination state t .
- At state 1, choose $u \in (0, 1]$ (the blackmail amount demanded) at a cost $-u$, and move to t with prob. u^2 , or stay in 1 with prob. $1 - u^2$.
- Every stationary policy is proper, but the **control set is not finite**.
- For any stationary μ with $\mu(1) = u$, we have

$$J_\mu(1) = -u + (1 - u^2)J_\mu(1)$$

from which $J_\mu(1) = -\frac{1}{u}$

- Thus $J^*(1) = -\infty$, and there is no optimal stationary policy.
- It turns out that **a nonstationary policy is optimal**: demand $\mu_k(1) = \gamma/(k + 1)$ at time k , with $\gamma \in (0, 1/2)$.
 - Blackmailer requests diminishing amounts over time, which add to ∞ .
 - The probability of the victim's refusal diminishes at a much faster rate, so the probability that the victim stays forever compliant is strictly positive.

SSP ALGORITHMS

- All the basic algorithms have counterparts
- “Easy” case: All policies proper, in which case the mappings T and T_μ are contractions
- Even in the general case all basic algorithms have satisfactory counterparts
- Value iteration, policy iteration
- Q-learning
- Optimistic policy iteration
- Asynchronous value iteration
- Asynchronous policy iteration
- See the text (Ch. 3, Vol. II) for details

6.231 DYNAMIC PROGRAMMING

LECTURE 19

LECTURE OUTLINE

- We begin a lecture series on approximate DP for large/intractable problems.
- Reading: Chapters 6 and 7, Vol. 2 (4th edition) of the text.
- Today we classify/overview the main approaches:
 - **Rollout/Simulation-based single policy iteration** (will not discuss this further)
 - **Approximation in value space** (approximate policy iteration, approximate VI, Q-Learning, Bellman error approach, approximate LP)
 - **Approximation in value space using problem approximation** (simplification - forms of aggregation - limited lookahead) - will not discuss much
 - **Approximation in policy space** (policy parametrization, gradient methods)

GENERAL ORIENTATION

- We will mainly adopt an n -state discounted model (the easiest case - but think of HUGE n).
- Extensions to SSP and average cost are possible (but more quirky). We will set aside for later.
- Other than manual/trial-and-error approaches (e.g., as in computer chess), the only other approaches are simulation-based. They are collectively known as “neuro-dynamic programming” or “reinforcement learning”.
- Simulation is essential for large state spaces because of its (potential) computational complexity **advantage in computing sums/expectations involving a very large number of terms**.
- Simulation also comes in handy when **an analytical model of the system is unavailable**, but a simulation/computer model is possible.
- Simulation-based methods are of three types:
 - Rollout (we will not discuss further)
 - Approximation in value space
 - Approximation in policy space

APPROXIMATION IN POLICY SPACE

- A brief discussion; we will return to it later.
- We parameterize the set of policies by a vector $r = (r_1, \dots, r_s)$ and we optimize the cost over r .
- Discounted problem example:
 - Each value of r defines a stationary policy, with cost starting at state i denoted by $J_i(r)$.
 - Use a gradient (or other) method to minimize over r

$$\bar{J}(r) = \sum_{i=1}^n q(i) J_i(r),$$

where $(q(1), \dots, q(n))$ is some probability distribution over the states.

- In a special case of this approach, the parameterization of the policies is indirect, through an approximate cost function.
 - A cost approximation architecture parameterized by r , defines a policy dependent on r via the minimization in Bellman's equation.

APPROX. IN VALUE SPACE - APPROACHES

- **Approximate PI** (policy eval./policy improve)
 - Uses simulation algorithms to approximate the cost J_μ of the current policy μ
 - Projected equation and aggregation approaches
- **Approximation of the optimal cost function J^***
 - **Approximate VI**: Approximate J_k in the VI $J_{k+1}(i) = (TJ_k)(i)$ with $\tilde{J}_k(r_k)$:

$$(\tilde{J}_{k+1}(r_{k+1}))(i) \approx (T\tilde{J}_k(r_k))(i)$$

- **Q-Learning**: Use simulation to approximate the optimal costs $J^*(i)$ or the Q -factors

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

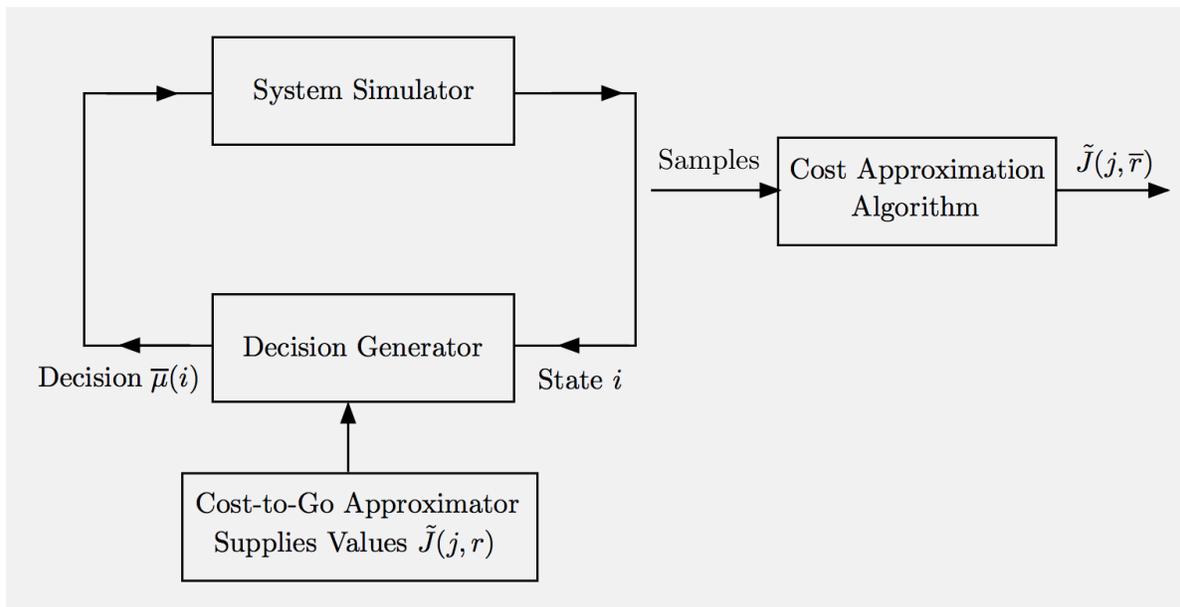
- **Bellman error approach**: Find r to

$$\min_r E_i \left\{ (\tilde{J}(i, r) - (T\tilde{J})(i, r))^2 \right\}$$

- **Approximate LP** (supplemented with clever schemes to overcome the large number of constraints issue)

POLICY EVALUATE/POLICY IMPROVE

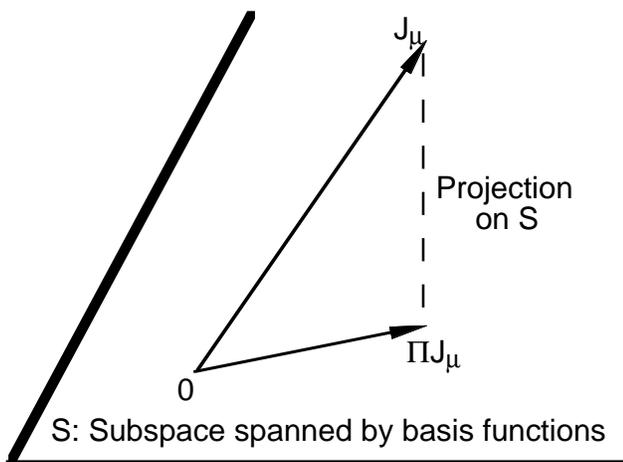
- General structure (sometimes referred to as **actor-critic** system)



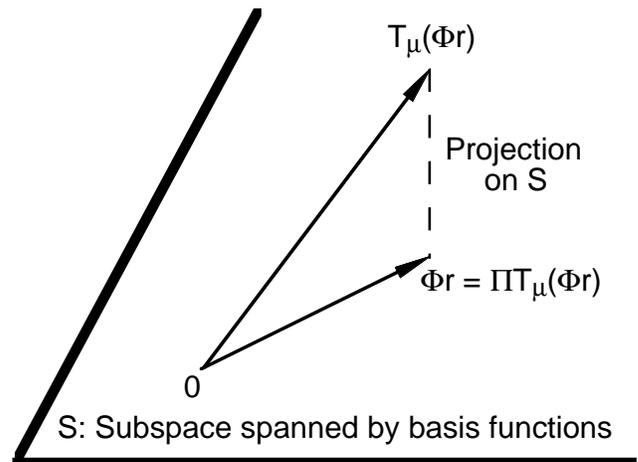
- $\tilde{J}(j, r)$ is the cost approximation for the preceding policy, used by the decision generator to compute the current policy $\bar{\mu}$ [whose cost is approximated by $\tilde{J}(j, \bar{r})$ using simulation]
- There are several cost approximation/policy evaluation algorithms
- There are several important issues relating to the design of each block (to be discussed in the future).

POLICY EVALUATION APPROACHES I

- Approximate the cost of the current policy by using a simulation method.
 - **Direct policy evaluation** - Cost samples generated by simulation, and optimization by least squares
 - **Indirect policy evaluation**
 - An example of indirect approach: Solving the **projected equation** $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm



Direct Method: Projection of cost vector J_μ



Indirect method: Solving a projected form of Bellman's equation

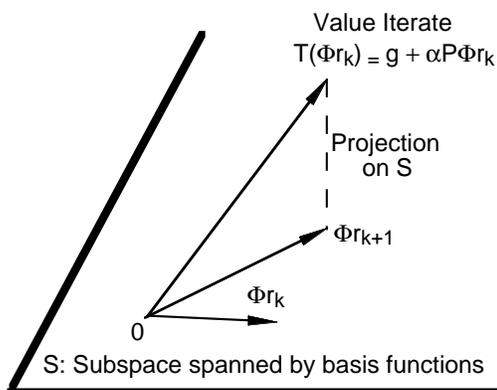
- Batch and incremental methods
- Regular and optimistic policy iteration

POLICY EVALUATION APPROACHES II

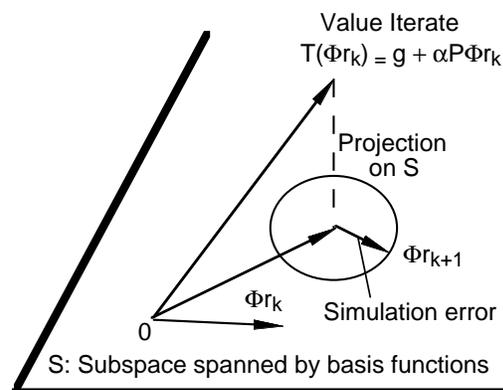
- **Projected equation methods**

- TD(λ): Stochastic iterative algorithm for solving $\Phi r = \Pi T_\mu(\Phi r)$
- LSPE(λ): A simulation-based form of *projected VI*

$$\Phi r_{k+1} = \Pi T_\mu(\Phi r_k) + \text{simulation noise}$$



Projected Value Iteration (PVI)



Least Squares Policy Evaluation (LSPE)

- LSTD(λ): Solves a simulation-based approximation using a linear system solver (e.g., Gaussian elimination/Matlab)

- **Aggregation:** (Related to projected Eqs) Solve

$$\Phi r = D T_\mu(\Phi r)$$

where the rows of D and Φ are prob. distributions (e.g., D and Φ “aggregate” rows and columns of the linear system $J = T_\mu J$).

THEORETICAL BASIS OF APPROXIMATE PI

- If policies are approximately evaluated using an approximation architecture:

$$\max_i |\tilde{J}(i, r_k) - J_{\mu^k}(i)| \leq \delta, \quad k = 0, 1, \dots$$

- If policy improvement is also approximate,

$$\max_i |(T_{\mu^{k+1}} \tilde{J})(i, r_k) - (T \tilde{J})(i, r_k)| \leq \epsilon, \quad k = 0, 1, \dots$$

- **Error Bound:** The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \rightarrow \infty} \max_i (J_{\mu^k}(i) - J^*(i)) \leq \frac{\epsilon + 2\alpha\delta}{(1 - \alpha)^2}$$

- Typical practical behavior: The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^* .

THE USE OF SIMULATION

- Simulation advantage: Compute (approximately) sums with a very large number of terms.

- **Example: Projection by Monte Carlo Simulation** Compute projection ΠJ of $J \in \mathfrak{R}^n$ on subspace $S = \{\Phi r \mid r \in \mathfrak{R}^s\}$, with respect to a weighted Euclidean norm $\|\cdot\|_v$.

- Find Φr^* , where

$$r^* = \arg \min_{r \in \mathfrak{R}^s} \|\Phi r - J\|_v^2 = \arg \min_{r \in \mathfrak{R}^s} \sum_{i=1}^n v_i (\phi(i)'r - J(i))^2$$

- Setting to 0 the gradient at r^* ,

$$r^* = \left(\sum_{i=1}^n v_i \phi(i) \phi(i)'\right)^{-1} \sum_{i=1}^n v_i \phi(i) J(i)$$

- Simulate the two “expected values” as follows

$$\hat{r}_k = \left(\sum_{t=1}^k \phi(i_t) \phi(i_t)'\right)^{-1} \sum_{t=1}^k \phi(i_t) J(i_t)$$

- Equivalent least squares alternative:

$$\hat{r}_k = \arg \min_{r \in \mathfrak{R}^s} \sum_{t=1}^k (\phi(i_t)'r - J(i_t))^2$$

THE ISSUE OF EXPLORATION

- To evaluate a policy μ , we need to generate cost samples using that policy - this biases the simulation by underrepresenting states that are unlikely to occur under μ .
- As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate.
- This seriously impacts the improved policy $\bar{\mu}$.
- This is known as **inadequate exploration** - a particularly acute difficulty when the randomness embodied in the transition probabilities is “relatively small” (e.g., a deterministic system).
- **One possibility** to guarantee adequate exploration: Frequently restart the simulation and ensure that the initial states employed form a rich and representative subset.
- **Another possibility**: Occasionally generating transitions that use a randomly selected control rather than the one dictated by the policy μ .
- **Related methods**, to be discussed later, use two Markov chains (one is the chain of the policy and is used to generate the transition sequence, the other is used to generate the state sequence).

APPROXIMATING Q-FACTORS

- The approach described so far for policy evaluation requires calculating expected values for all controls $u \in U(i)$ (and knowledge of $p_{ij}(u)$).
- **Model-free alternative:** Approximate Q -factors

$$\tilde{Q}(i, u, r) \approx \sum_{j=1}^n p_{ij}(u) (g(i, u, j) + \alpha J_{\mu}(j))$$

and use for policy improvement the minimization

$$\bar{\mu}(i) = \arg \min_{u \in U(i)} \tilde{Q}(i, u, r)$$

- r is an adjustable parameter vector and $\tilde{Q}(i, u, r)$ is a parametric architecture, such as

$$\tilde{Q}(i, u, r) = \sum_{k=1}^m r_k \phi_k(i, u)$$

- Can use any method for constructing cost approximations, e.g., TD(λ).
- Use the Markov chain with states (i, u) - $p_{ij}(\mu(i))$ is the transition prob. to $(j, \mu(i))$, 0 to other (j, u') .
- **Major concern:** Acutely diminished exploration.

6.231 DYNAMIC PROGRAMMING

LECTURE 20

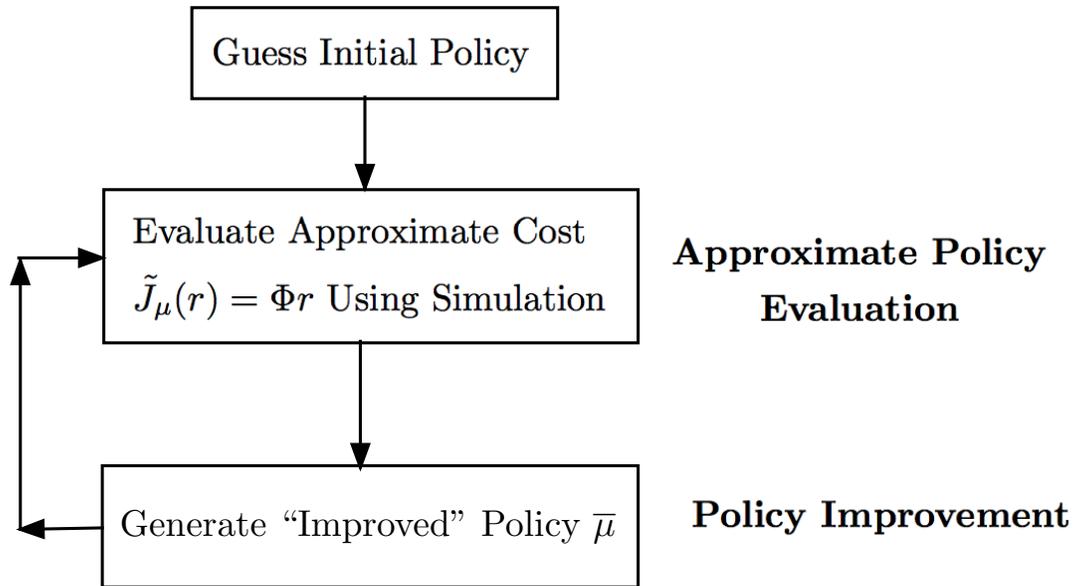
LECTURE OUTLINE

- Discounted problems - Approximation on subspace $\{\Phi r \mid r \in \mathbb{R}^s\}$
- Approximate policy evaluation/policy improvement
- Direct approach - Least squares
- Approximate VI based on the direct approach
- Indirect approach - The projected equation
- Contraction properties - Error bounds
- Matrix form of the projected equation
- Simulation-based implementation
- LSTD and LSPE methods

REVIEW: APPROXIMATION IN VALUE SPACE

- Finite-spaces discounted problems: Defined by mappings T_μ and T ($TJ = \min_\mu T_\mu J$).
- **Exact methods:**
 - VI: $J_{k+1} = TJ_k$
 - PI: $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$, $T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$
 - LP: $\min_J c'J$ subject to $J \leq TJ$
- **Approximate versions:** Plug-in subspace approximation with Φr in place of J
 - VI: $\Phi r_{k+1} \approx T\Phi r_k$
 - PI: $\Phi r_k \approx T_{\mu^k} \Phi r_k$, $T_{\mu^{k+1}} \Phi r_k = T\Phi r_k$
 - LP: $\min_r c'\Phi r$ subject to $\Phi r \leq T\Phi r$
- Approximation onto subspace $S = \{\Phi r \mid r \in \mathbb{R}^s\}$ is often done by **projection** with respect to (weighted) Euclidean norm.
- Another possibility is **aggregation**.

APPROXIMATE PI



- Linear cost function approximation

$$\tilde{J}(r) = \Phi r$$

where Φ is full rank $n \times s$ matrix with columns the basis functions, and i th row denoted $\phi(i)'$.

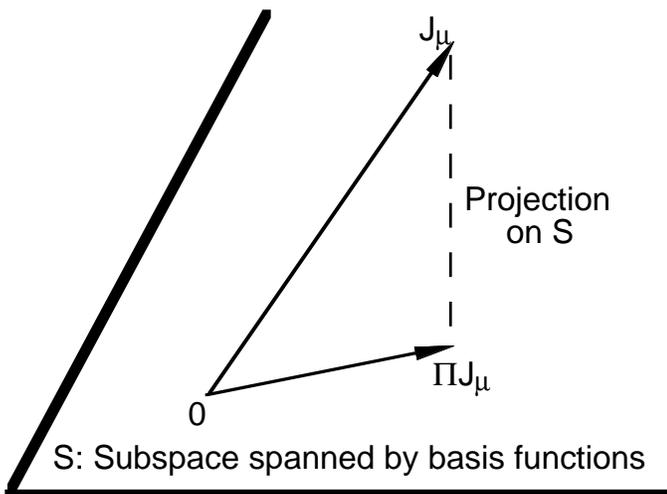
- Policy "improvement"

$$\bar{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u) (g(i, u, j) + \alpha \phi(j)'r)$$

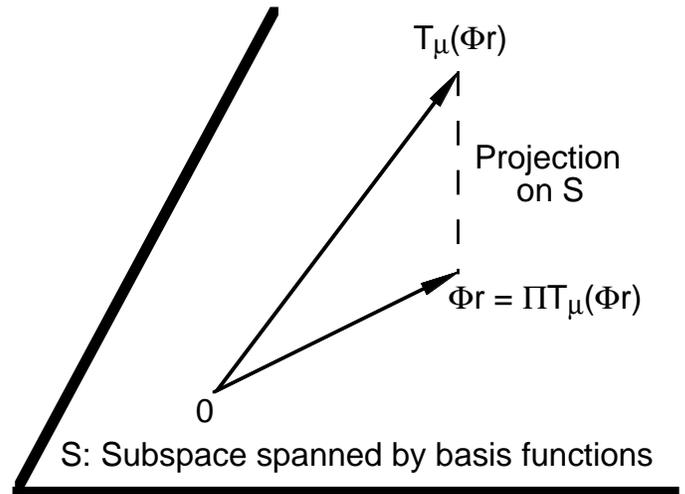
- Theoretical basis is the error bound given in preceding lectures

POLICY EVALUATION

- Focus on policy evaluation: approximate the cost of the current policy by using a simulation method.
 - **Direct policy evaluation** - Cost samples generated by simulation, and optimization by least squares
 - **Indirect policy evaluation** - solving the projected equation $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm



Direct Method: Projection of cost vector J_μ



Indirect method: Solving a projected form of Bellman's equation

- Recall that projection can be implemented by simulation and least squares

DIRECT POLICY EVALUATION

- Suppose we can implement in a simulator the current policy μ , and want to calculate J_μ by simulation.
- **Generate by simulation sample costs.** Then:

$$J_\mu(i) \approx \frac{1}{M_i} \sum_{m=1}^{M_i} c(i, m)$$

$c(i, m)$: m th (noisy) sample cost starting from state i

- Use an approximation architecture $\tilde{J}_\mu(i, r)$, where r is a tunable parameter vector.
- **Direct approach:** Calculate an optimal value r^* of r by a least squares fit

$$r^* = \arg \min_r \sum_{i=1}^n \sum_{m=1}^{M_i} (c(i, m) - \tilde{J}_\mu(i, r))^2$$

- Note that this is much easier when the architecture is linear - but this is not a requirement.

IMPLEMENTATION OF DIRECT APPROACH

- Generate batch: an N -transition portion (i_0, \dots, i_N) of a simulated trajectory (no model needed)
- We view the numbers

$$\sum_{t=k}^{N-1} \alpha^{t-k} g(i_t, \mu(i_t), i_{t+1}), \quad k = 0, \dots, N-1,$$

as cost samples, one per initial state i_0, \dots, i_{N-1}

- **Solve the least squares problem**

$$\min_r \frac{1}{2} \sum_{k=0}^{N-1} \left(\tilde{J}(i_k, r) - \sum_{t=k}^{N-1} \alpha^{t-k} g(i_t, \mu(i_t), i_{t+1}) \right)^2$$

- If $\tilde{J}(i_k, r)$ is linear in r , this problem can be solved by matrix inversion.
- **Another possibility: Use a gradient-type method** [applies also to cases where $\tilde{J}(i_k, r)$ is nonlinear]
- A variety of gradient methods: Batch and incremental

DIRECT APPROACH FOR APPROXIMATE VI

- Approximates sequentially $J_k(i) = (T^k J_0)(i)$, $k = 1, 2, \dots$, with $\tilde{J}_k(i, r_k)$, e.g., $\tilde{J}_k = \Phi r_k$
- The starting function J_0 is given (e.g., $J_0 \equiv 0$)
- After a large enough number N of steps, $\tilde{J}_N(i, r_N)$ is used as approximation $\tilde{J}(i, r)$ to $J^*(i)$
- **Fitted Value Iteration:** A sequential “fit” to produce \tilde{J}_{k+1} from \tilde{J}_k , i.e., $\tilde{J}_{k+1} \approx T \tilde{J}_k$ or (for a single policy μ) $\tilde{J}_{k+1} \approx T_\mu \tilde{J}_k$
 - For a “small” subset S_k of states i , compute

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

- “Fit” the function $\tilde{J}_{k+1}(i, r_{k+1})$ to the “small” set of values $(T \tilde{J}_k)(i)$, $i \in S_k$
- **Error Bound:** If the fit is uniformly accurate within $\delta > 0$ (i.e., $\max_i |\tilde{J}_{k+1}(i) - T \tilde{J}_k(i)| \leq \delta$),

$$\limsup_{k \rightarrow \infty} \max_{i=1, \dots, n} (\tilde{J}_k(i, r_k) - J^*(i)) \leq \frac{2\alpha\delta}{(1-\alpha)^2}$$

AN EXAMPLE OF FAILURE

- Consider two states 1 and 2, and a single policy.
 - Deterministic transitions: $1 \rightarrow 2$ and $2 \rightarrow 2$
 - Transition costs $\equiv 0$, so $J^*(1) = J^*(2) = 0$.

• Consider approximate VI scheme that approximates cost functions in $S = \{(r, 2r) \mid r \in \mathfrak{R}\}$ with a weighted least squares fit; here $\Phi = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

- Given $J_k = (r_k, 2r_k)$, we find $J_{k+1} = (r_{k+1}, 2r_{k+1})$, where for weights $v_1, v_2 > 0$, r_{k+1} is obtained as

$$r_{k+1} = \arg \min_r \left[v_1 (r - (TJ_k)(1))^2 + v_2 (2r - (TJ_k)(2))^2 \right]$$

or $\Phi r_{k+1} = \Pi_v T(\Phi r_k)$ (= least squares fit of $T(\Phi r_k)$).

- With straightforward calculation

$$r_{k+1} = \alpha \beta r_k, \quad \text{where } \beta = 2(v_1 + 2v_2)/(v_1 + 4v_2) > 1$$

- So if $\alpha > 1/\beta$, the sequence $\{r_k\}$ diverges and so does $\{J_k\}$.

- **Difficulty:** T is a contraction, but $\Pi_v T$ is not

- **Norm mismatch problem** (to be reencountered)

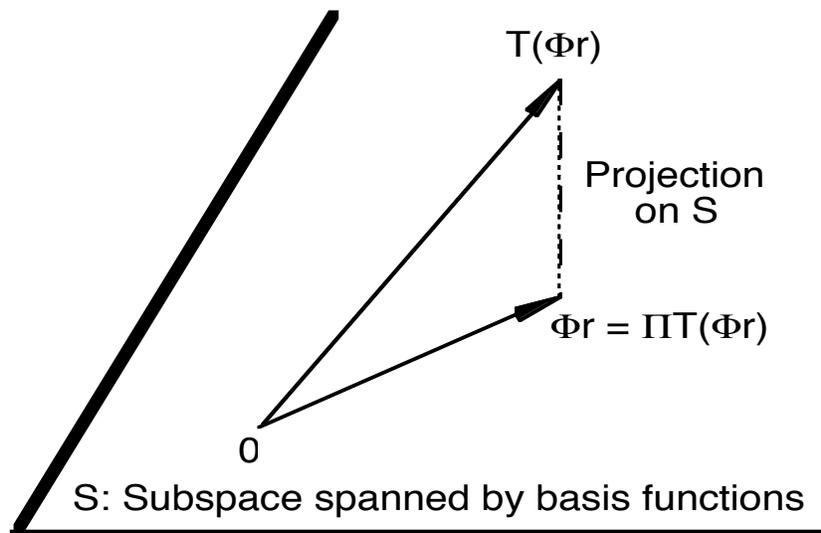
INDIRECT POLICY EVALUATION

- For the current policy μ , consider the corresponding mapping T :

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

- The solution J_μ of Bellman's equation $J = TJ$ is approximated by the solution of

$$\Phi r = \Pi T(\Phi r)$$



Indirect method: Solving a projected form of Bellman's equation

WEIGHTED EUCLIDEAN PROJECTIONS

- Consider a weighted Euclidean norm

$$\|J\|_v = \sqrt{\sum_{i=1}^n v_i (J(i))^2},$$

where v is a vector of positive weights v_1, \dots, v_n .

- Let Π denote the projection operation onto

$$S = \{\Phi r \mid r \in \mathbb{R}^s\}$$

with respect to this norm, i.e., for any $J \in \mathbb{R}^n$,

$$\Pi J = \Phi r^*$$

where

$$r^* = \arg \min_{r \in \mathbb{R}^s} \|J - \Phi r\|_v^2$$

KEY QUESTIONS AND RESULTS

- Does the projected equation have a solution?
- Under what conditions is the mapping ΠT a contraction, so ΠT has unique fixed point?
- Assuming ΠT has unique fixed point Φr^* , how close is Φr^* to J_μ ?
- **Assumption:** P has a single recurrent class and no transient states, i.e., it has steady-state probabilities that are positive

$$\xi_j = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=1}^N P(i_k = j \mid i_0 = i) > 0$$

- **Proposition:** ΠT is contraction of modulus α with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where $\xi = (\xi_1, \dots, \xi_n)$ is the steady-state probability vector. The unique fixed point Φr^* of ΠT satisfies

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1 - \alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi$$

PRELIMINARIES: PROJECTION PROPERTIES

- Important property of the projection Π on S with weighted Euclidean norm $\|\cdot\|_v$. For all $J \in \mathbb{R}^n$, $\bar{J} \in S$, the **Pythagorean Theorem** holds:

$$\|J - \bar{J}\|_v^2 = \|J - \Pi J\|_v^2 + \|\Pi J - \bar{J}\|_v^2$$

Proof: Geometrically, $(J - \Pi J)$ and $(\Pi J - \bar{J})$ are orthogonal in the scaled geometry of the norm $\|\cdot\|_v$, where two vectors $x, y \in \mathbb{R}^n$ are orthogonal if $\sum_{i=1}^n v_i x_i y_i = 0$. Expand the quadratic in the RHS below:

$$\|J - \bar{J}\|_v^2 = \|(J - \Pi J) + (\Pi J - \bar{J})\|_v^2$$

- The Pythagorean Theorem implies that the **projection is nonexpansive**, i.e.,

$$\|\Pi J - \Pi \bar{J}\|_v \leq \|J - \bar{J}\|_v, \quad \text{for all } J, \bar{J} \in \mathbb{R}^n.$$

To see this, note that

$$\begin{aligned} \|\Pi(J - \bar{J})\|_v^2 &\leq \|\Pi(J - \bar{J})\|_v^2 + \|(I - \Pi)(J - \bar{J})\|_v^2 \\ &= \|J - \bar{J}\|_v^2 \end{aligned}$$

PROOF OF CONTRACTION PROPERTY

- **Lemma:** We have

$$\|Pz\|_{\xi} \leq \|z\|_{\xi}, \quad z \in \mathfrak{R}^n$$

Proof: Let p_{ij} be the components of P . For all $z \in \mathfrak{R}^n$, we have

$$\begin{aligned} \|Pz\|_{\xi}^2 &= \sum_{i=1}^n \xi_i \left(\sum_{j=1}^n p_{ij} z_j \right)^2 \leq \sum_{i=1}^n \xi_i \sum_{j=1}^n p_{ij} z_j^2 \\ &= \sum_{j=1}^n \sum_{i=1}^n \xi_i p_{ij} z_j^2 = \sum_{j=1}^n \xi_j z_j^2 = \|z\|_{\xi}^2, \end{aligned}$$

where the inequality follows from the convexity of the quadratic function, and the next to last equality follows from the defining property $\sum_{i=1}^n \xi_i p_{ij} = \xi_j$ of the steady-state probabilities.

- Using the lemma, the nonexpansiveness of Π , and the definition $TJ = g + \alpha PJ$, we have

$$\|\Pi TJ - \Pi T\bar{J}\|_{\xi} \leq \|TJ - T\bar{J}\|_{\xi} = \alpha \|P(J - \bar{J})\|_{\xi} \leq \alpha \|J - \bar{J}\|_{\xi}$$

for all $J, \bar{J} \in \mathfrak{R}^n$. Hence ΠT is a contraction of modulus α .

PROOF OF ERROR BOUND

- Let Φr^* be the fixed point of ΠT . We have

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1 - \alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi.$$

Proof: We have

$$\begin{aligned} \|J_\mu - \Phi r^*\|_\xi^2 &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \|\Pi J_\mu - \Phi r^*\|_\xi^2 \\ &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \|\Pi T J_\mu - \Pi T(\Phi r^*)\|_\xi^2 \\ &\leq \|J_\mu - \Pi J_\mu\|_\xi^2 + \alpha^2 \|J_\mu - \Phi r^*\|_\xi^2, \end{aligned}$$

where

- The first equality uses the Pythagorean Theorem
- The second equality holds because J_μ is the fixed point of T and Φr^* is the fixed point of ΠT
- The inequality uses the contraction property of ΠT .

Q.E.D.

MATRIX FORM OF PROJECTED EQUATION

- Its solution is the vector $J = \Phi r^*$, where r^* solves the problem

$$\min_{r \in \mathbb{R}^s} \left\| \Phi r - (g + \alpha P \Phi r^*) \right\|_{\xi}^2.$$

- Setting to 0 the gradient with respect to r of this quadratic, we obtain

$$\Phi' \Xi (\Phi r^* - (g + \alpha P \Phi r^*)) = 0,$$

where Ξ is the diagonal matrix with the steady-state probabilities ξ_1, \dots, ξ_n along the diagonal.

- This is just the **orthogonality condition**: The error $\Phi r^* - (g + \alpha P \Phi r^*)$ is “orthogonal” to the subspace spanned by the columns of Φ .
- Equivalently,

$$C r^* = d,$$

where

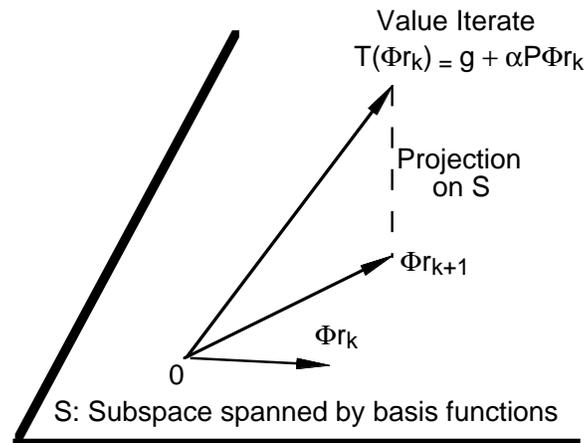
$$C = \Phi' \Xi (I - \alpha P) \Phi, \quad d = \Phi' \Xi g.$$

PROJECTED EQUATION: SOLUTION METHODS

- **Matrix inversion:** $r^* = C^{-1}d$
- **Projected Value Iteration (PVI) method:**

$$\Phi r_{k+1} = \Pi T(\Phi r_k) = \Pi(g + \alpha P \Phi r_k)$$

Converges to r^* because ΠT is a contraction.



- PVI can be written as:

$$r_{k+1} = \arg \min_{r \in \mathbb{R}^s} \left\| \Phi r - (g + \alpha P \Phi r_k) \right\|_{\xi}^2$$

By setting to 0 the gradient with respect to r ,

$$\Phi' \Xi (\Phi r_{k+1} - (g + \alpha P \Phi r_k)) = 0,$$

which yields

$$r_{k+1} = r_k - (\Phi' \Xi \Phi)^{-1} (C r_k - d)$$

SIMULATION-BASED IMPLEMENTATIONS

- **Key idea:** Calculate simulation-based approximations based on k samples

$$C_k \approx C, \quad d_k \approx d$$

- Matrix inversion $r^* = C^{-1}d$ is approximated by

$$\hat{r}_k = C_k^{-1}d_k$$

This is the **LSTD** (Least Squares Temporal Differences) Method.

- PVI method $r_{k+1} = r_k - (\Phi' \Xi \Phi)^{-1} (C r_k - d)$ is approximated by

$$r_{k+1} = r_k - G_k (C_k r_k - d_k)$$

where

$$G_k \approx (\Phi' \Xi \Phi)^{-1}$$

This is the **LSPE** (Least Squares Policy Evaluation) Method.

- **Key fact:** C_k , d_k , and G_k can be computed with low-dimensional linear algebra (of order s ; the number of basis functions).

SIMULATION MECHANICS

- We generate an infinitely long trajectory (i_0, i_1, \dots) of the Markov chain, so states i and transitions (i, j) appear with long-term frequencies ξ_i and p_{ij} .
- After generating the transition (i_t, i_{t+1}) , we compute the row $\phi(i_t)'$ of Φ and the cost component $g(i_t, i_{t+1})$.
- We form

$$C_k = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) (\phi(i_t) - \alpha \phi(i_{t+1}))' \approx \Phi' \Xi (I - \alpha P) \Phi$$

$$d_k = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) g(i_t, i_{t+1}) \approx \Phi' \Xi g$$

Also in the case of LSPE

$$G_k = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) \phi(i_t)' \approx \Phi' \Xi \Phi$$

- Convergence proof: **View C , d , and G as expectations; use law of large numbers arguments.**
- Note that C_k , d_k , and G_k can be formed incrementally.

6.231 DYNAMIC PROGRAMMING

LECTURE 21

LECTURE OUTLINE

- Review of approximate policy iteration
- Projected equation methods for policy evaluation
- Optimistic versions
- Multistep projected equation methods
- Bias-variance tradeoff
- Exploration-enhanced implementations
- Oscillations

REVIEW: PROJECTED BELLMAN EQUATION

- For a fixed policy μ to be evaluated, consider the corresponding mapping T :

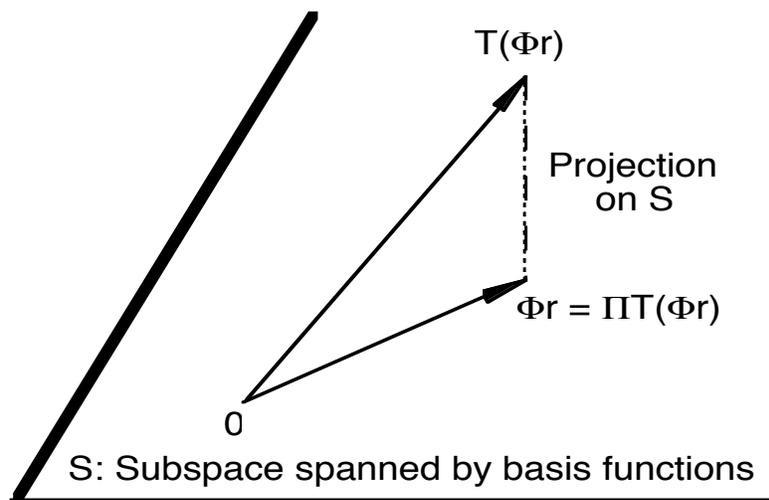
$$(TJ)(i) = \sum_{j=1}^n p_{ij} (g(i, j) + \alpha J(j)), \quad i = 1, \dots, n,$$

or more compactly,

$$TJ = g + \alpha PJ$$

- Approximate Bellman's equation $J = TJ$ by $\Phi r = \Pi T(\Phi r)$ or the matrix form/orthogonality condition $Cr^* = d$, where

$$C = \Phi' \Xi (I - \alpha P) \Phi, \quad d = \Phi' \Xi g.$$



Indirect method: Solving a projected form of Bellman's equation

PROJECTED EQUATION METHODS

- **Matrix inversion:** $r^* = C^{-1}d$
- **Iterative Projected Value Iteration (PVI) method:**

$$\Phi r_{k+1} = \Pi T(\Phi r_k) = \Pi(g + \alpha P \Phi r_k)$$

Converges to r^* if ΠT is a contraction. True if Π is projection w.r.t. steady-state distribution norm.

- **Simulation-Based Implementations:** Generate $k+1$ simulated transitions sequence $\{i_0, i_1, \dots, i_k\}$ and approximations $C_k \approx C$ and $d_k \approx d$:

$$C_k = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) (\phi(i_t) - \alpha \phi(i_{t+1}))' \approx \Phi' \Xi (I - \alpha P) \Phi$$

$$d_k = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) g(i_t, i_{t+1}) \approx \Phi' \Xi g$$

- **LSTD:** $\hat{r}_k = C_k^{-1} d_k$
- **LSPE:** $r_{k+1} = r_k - G_k (C_k r_k - d_k)$ where $G_k \approx (\Phi' \Xi \Phi)^{-1}$. Converges to r^* if ΠT is contraction.
- **Key fact:** C_k , d_k , and G_k can be computed with low-dimensional linear algebra (of order s ; the number of basis functions).

OPTIMISTIC VERSIONS

- Use coarse approximations $C_k \approx C$ and $d_k \approx d$, based on few simulation samples
- **PI context:** Evaluate (coarsely) current policy μ , then do a policy improvement
- Very complex behavior (see the subsequent discussion on oscillations)
- Often approaches the limit more quickly (as optimistic methods often do)
- The matrix inversion/LSTD method has serious problems due to large simulation noise (because of limited sampling)
- A stepsize $\gamma \in (0, 1]$ in LSPE may be useful to damp the effect of simulation noise:

$$r_{k+1} = r_k - \gamma G_k (C_k r_k - d_k)$$

- In the context of PI, **LSPE tends to cope better because of its iterative nature** (when a policy is changed, the current r_k , C_k , d_k , G_k may be used as a “hot start” for the iterations of the new policy evaluation)

MULTISTEP METHODS FOR POL. EVALUATION

- Introduce a multistep version of Bellman's equation $J = T^{(\lambda)}J$, where for $\lambda \in [0, 1)$,

$$T^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

- $T^{(\lambda)}$ has the same fixed point as T , so it may be used as basis for approximation.
- T^t is a contraction with modulus α^t , with respect to the weighted Euclidean norm $\|\cdot\|_{\xi}$, where ξ is the steady-state distribution vector of the chain.
- Hence $T^{(\lambda)}$ is a contraction with modulus

$$\alpha_{\lambda} = (1 - \lambda) \sum_{t=0}^{\infty} \alpha^{t+1} \lambda^t = \frac{\alpha(1 - \lambda)}{1 - \alpha\lambda}$$

Note that $\alpha_{\lambda} \rightarrow 0$ as $\lambda \rightarrow 1$.

- Let Φr_{λ}^* be the fixed point of $\Pi T^{(\lambda)}$. Then

$$\|J_{\mu} - \Phi r_{\lambda}^*\|_{\xi} \leq \frac{1}{\sqrt{1 - \alpha_{\lambda}^2}} \|J_{\mu} - \Pi J_{\mu}\|_{\xi}$$

- Φr_{λ}^* depends on λ ; is closer to ΠJ_{μ} as $\lambda \rightarrow 1$.

BIAS-VARIANCE TRADEOFF

- Error bound $\|J_\mu - \Phi r_\lambda^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha_\lambda^2}} \|J_\mu - \Pi J_\mu\|_\xi$

- **Bias-variance tradeoff:**

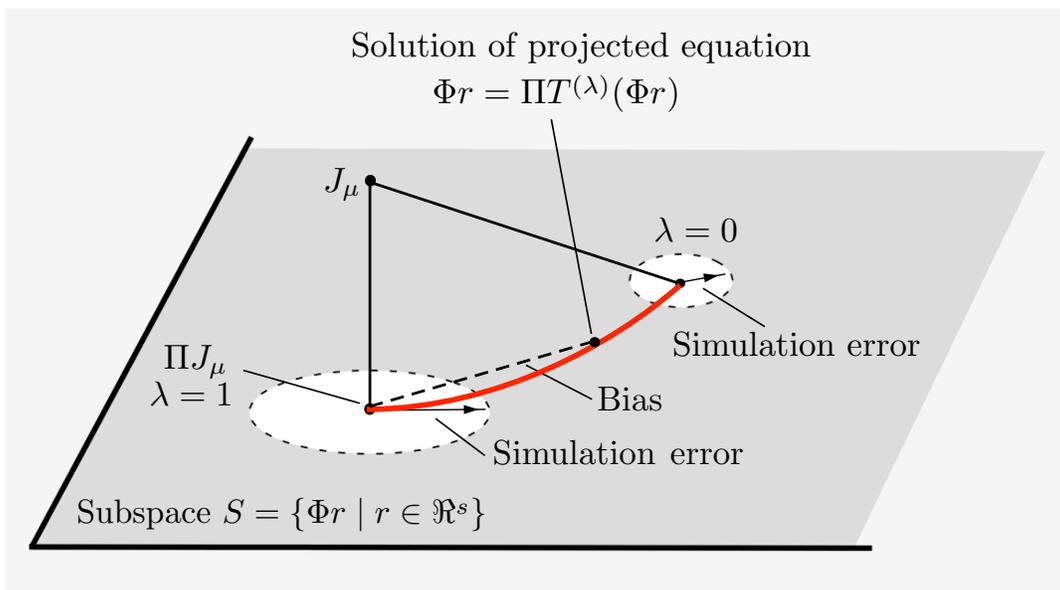
- As $\lambda \uparrow 1$, we have $\alpha_\lambda \downarrow 0$, so error bound (and the quality of approximation) improves as $\lambda \uparrow 1$. In fact

$$\lim_{\lambda \uparrow 1} \Phi r_\lambda^* = \Pi J_\mu \quad (= \text{the direct approx. solution})$$

- But simulation noise in approximating

$$T^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

increases.



MORE ON MULTISTEP METHODS

- The simulation process to obtain $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ is similar to the case $\lambda = 0$ (single simulation trajectory i_0, i_1, \dots more complex formulas)

$$C_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) \sum_{m=t}^k \alpha^{m-t} \lambda^{m-t} (\phi(i_m) - \alpha \phi(i_{m+1}))'$$

$$d_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^k \phi(i_t) \sum_{m=t}^k \alpha^{m-t} \lambda^{m-t} g_{i_m}$$

- In the context of approximate policy iteration, we can use optimistic versions (few samples between policy updates).
- Many different/incremental versions ... see text.
- Note the **λ -tradeoffs**:
 - As $\lambda \uparrow 1$, $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ contain more “simulation noise”, so more samples are needed for a close approximation of r_λ (the solution of the projected equation)
 - The error bound $\|J_\mu - \Phi r_\lambda\|_\xi$ becomes smaller
 - As $\lambda \uparrow 1$, $\Pi T^{(\lambda)}$ becomes a contraction for **arbitrary** projection norm

POLICY ITERATION ISSUES - EXPLORATION

- **1st major issue: exploration.** Common remedy is the **off-policy approach:** Replace P of current policy with

$$\bar{P} = (I - B)P + BQ,$$

where B is a diagonal matrix with $\beta_i \in [0, 1]$ on the diagonal, and Q is another transition matrix.

- Then LSTD and LSPE formulas must be modified ... otherwise the policy associated with \bar{P} (not P) is evaluated (see the textbook, Section 6.4).
- Alternatives: **Geometric and free-form sampling**
- Both of these use multiple short simulated trajectories, with random restart state, chosen to enhance exploration
- Geometric sampling uses trajectories with geometrically distributed number of transitions with parameter $\lambda \in [0, 1)$. It implements LSTD(λ) and LSPE(λ) with exploration (see the text).
- Free-form sampling uses trajectories with more generally distributed number of transitions. It implements method for approximation of the solution of a generalized multistep Bellman equation (see the text).

POLICY ITERATION ISSUES - OSCILLATIONS

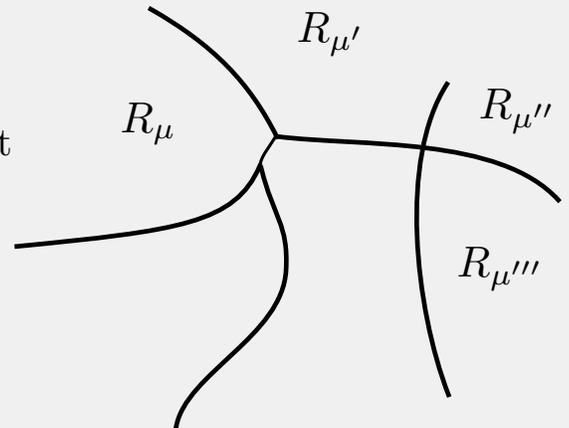
- Define for each policy μ

$$R_\mu = \{r \mid T_\mu(\Phi r) = T(\Phi r)\}$$

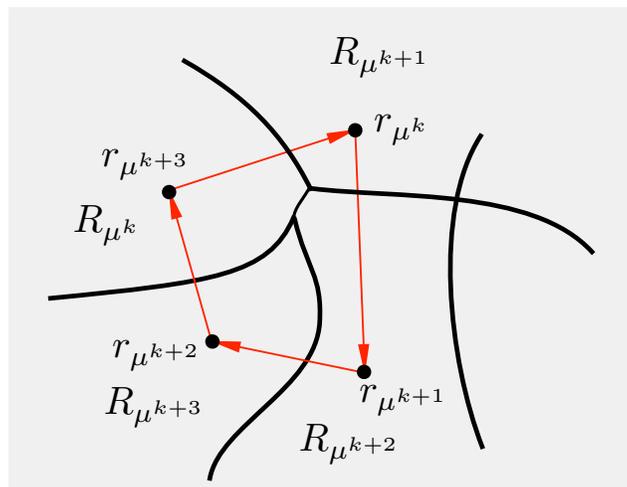
- These sets form the **greedy partition** of the parameter r -space

$$R_\mu = \{r \mid T_\mu(\Phi r) = T(\Phi r)\}$$

For a policy μ , R_μ is the set of all r such that policy improvement based on Φr produces μ

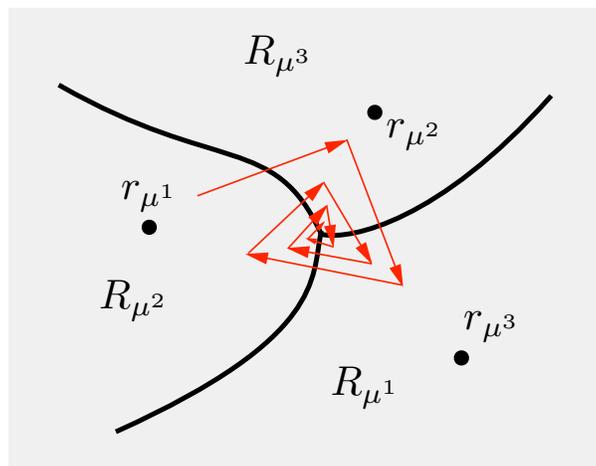


- Oscillations of nonoptimistic approx.: r_μ is generated by an evaluation method so that $\Phi r_\mu \approx J_\mu$



MORE ON OSCILLATIONS/CHATTERING

- For optimistic PI a different picture holds



- Oscillations are less violent, but the “limit” point is meaningless!
- Fundamentally, oscillations are due to the **lack of monotonicity of the projection operator**, i.e., $J \leq J'$ does not imply $\Pi J \leq \Pi J'$.
- If approximate PI uses policy evaluation

$$\Phi r = (WT_{\mu})(\Phi r)$$

with W some monotone operator, the generated policies converge (to a possibly nonoptimal limit).

- **The operator W used in the aggregation approach has this monotonicity property.**

6.231 DYNAMIC PROGRAMMING

LECTURE 22

LECTURE OUTLINE

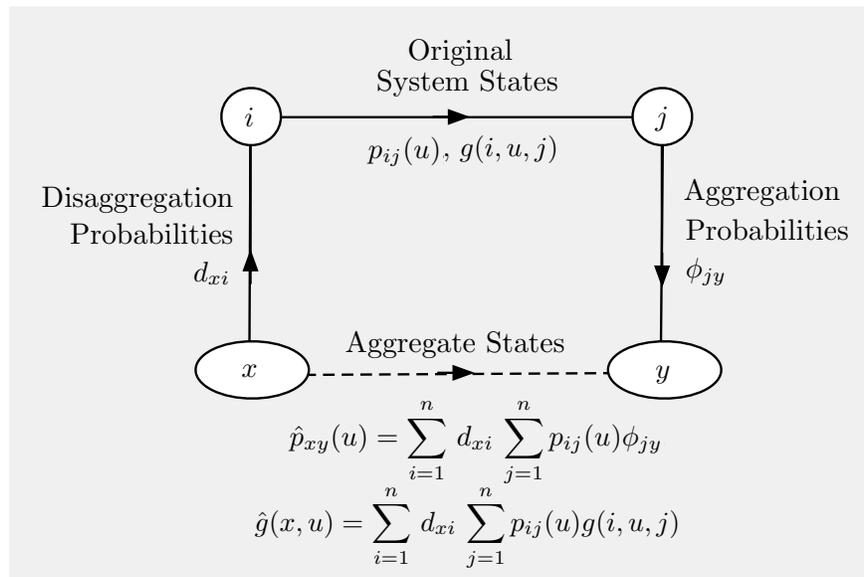
- Aggregation as an approximation methodology
- Aggregate problem
- Examples of aggregation
- Simulation-based aggregation
- Q-Learning

PROBLEM APPROXIMATION - AGGREGATION

- Another major idea in ADP is to approximate the cost-to-go function of the problem with the cost-to-go function of a simpler problem. The simplification is often ad-hoc/problem dependent.
- Aggregation is a systematic approach for problem approximation. Main elements:
 - Introduce a few “aggregate” states, viewed as the states of an “aggregate” system
 - Define transition probabilities and costs of the aggregate system, by relating original system states with aggregate states
 - Solve (exactly or approximately) the “aggregate” problem by any kind of value or policy iteration method (including simulation-based methods)
 - Use the optimal cost of the aggregate problem to approximate the optimal cost of the original problem
- Hard aggregation example: Aggregate states are subsets of original system states, treated as if they all have the same cost.

AGGREGATION/DISAGGREGATION PROBS

- The aggregate system transition probabilities are defined via two (somewhat arbitrary) choices



- For each original system state j and aggregate state y , the **aggregation probability** ϕ_{jy}
 - The “degree of membership of j in the aggregate state y .”
 - In hard aggregation, $\phi_{jy} = 1$ if state j belongs to aggregate state/subset y .
- For each aggregate state x and original system state i , the **disaggregation probability** d_{xi}
 - The “degree of i being representative of x .”
 - In hard aggregation, one possibility is all states i that belongs to aggregate state/subset x have equal d_{xi} .

AGGREGATE PROBLEM

- The transition probability from aggregate state x to aggregate state y under control u

$$\hat{p}_{xy}(u) = \sum_{i=1}^n d_{xi} \sum_{j=1}^n p_{ij}(u) \phi_{jy}, \quad \text{or } \hat{P}(u) = DP(u)\Phi$$

where the rows of D and Φ are the disaggr. and aggr. probs.

- The aggregate expected transition cost is

$$\hat{g}(x, u) = \sum_{i=1}^n d_{xi} \sum_{j=1}^n p_{ij}(u) g(i, u, j), \quad \text{or } \hat{g} = DPg$$

- The optimal cost function of the aggregate problem, denoted \hat{R} , is

$$\hat{R}(x) = \min_{u \in U} \left[\hat{g}(x, u) + \alpha \sum_y \hat{p}_{xy}(u) \hat{R}(y) \right], \quad \forall x$$

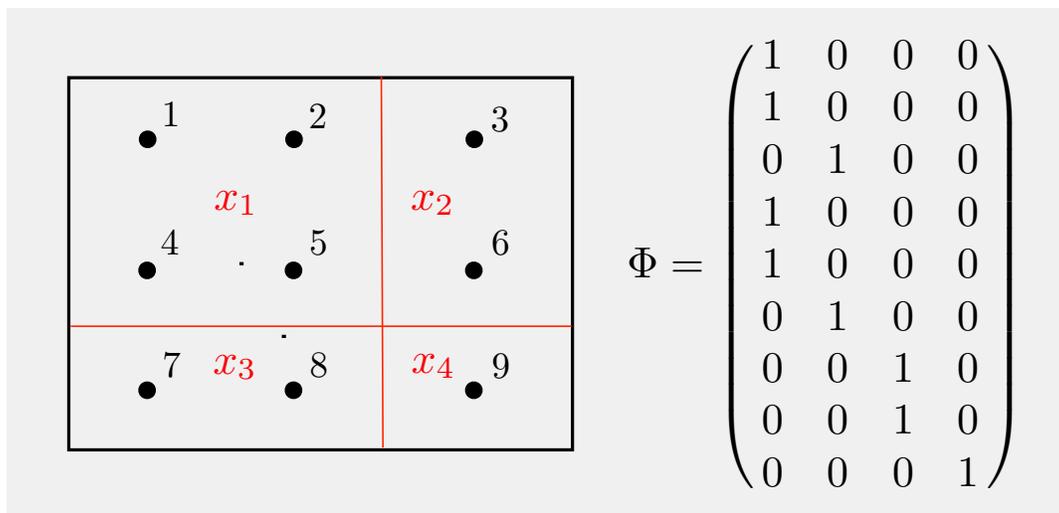
or $\hat{R} = \min_u [\hat{g} + \alpha \hat{P} \hat{R}]$ - Bellman's equation for the aggregate problem.

- The optimal cost function J^* of the original problem is approximated using interpolation:

$$\tilde{J}(j) = \sum_y \phi_{jy} \hat{R}(y), \quad \forall j$$

EXAMPLE I: HARD AGGREGATION

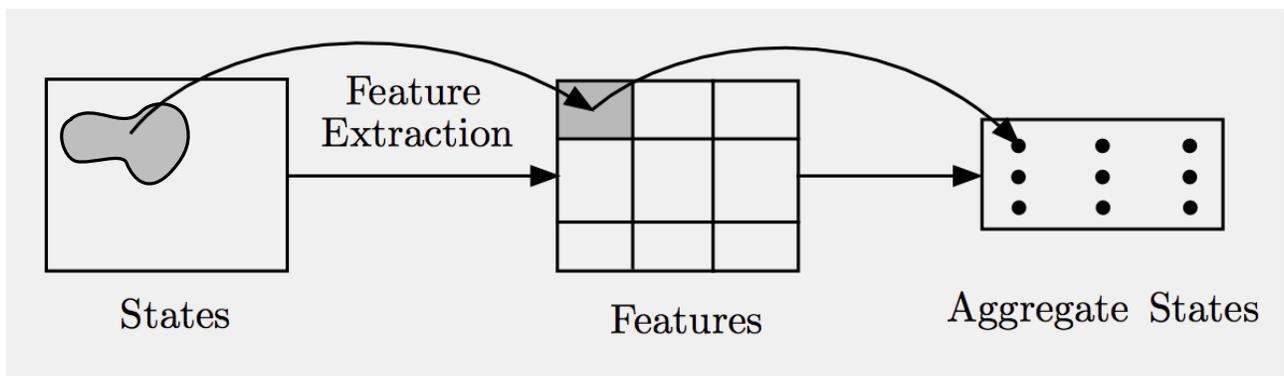
- Group the original system states into subsets, and view each subset as an aggregate state
- **Aggregation probs:** $\phi_{jy} = 1$ if j belongs to aggregate state y .



- **Disaggregation probs:** There are many possibilities, e.g., all states i within aggregate state x have equal prob. d_{xi} .
- **If optimal cost vector J^* is piecewise constant over the aggregate states/subsets, hard aggregation is exact.** Suggests grouping states with “roughly equal” cost into aggregates.
- **Soft aggregation** (provides “soft boundaries” between aggregate states).

EXAMPLE II: FEATURE-BASED AGGREGATION

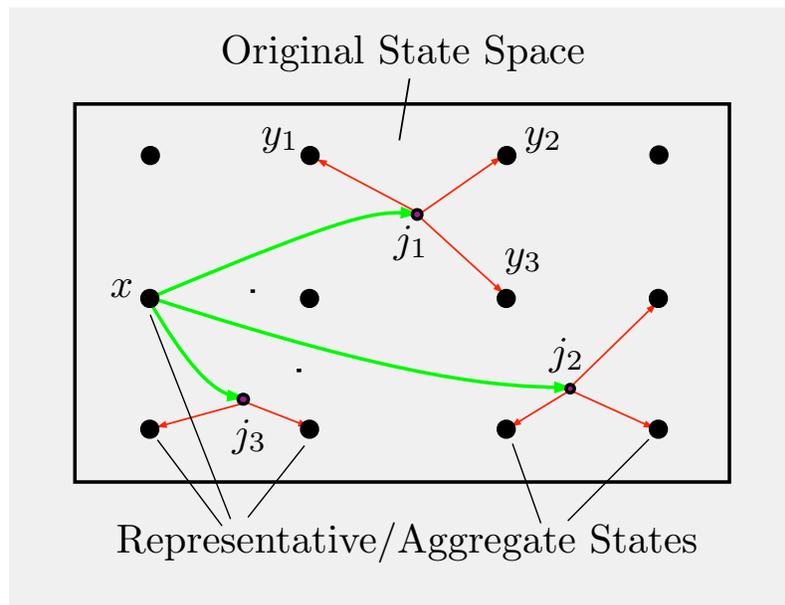
- If we know good features, it makes sense to **group together states that have “similar features”**
- Essentially discretize the features and assign a weight to each discretization point



- A general approach for passing from a feature-based state representation to an aggregation-based architecture
- Hard aggregation architecture based on features is more powerful (**nonlinear/piecewise constant in the features, rather than linear**)
- ... but may require many more aggregate states to reach the same level of performance as the corresponding linear feature-based architecture

EXAMPLE III: REP. STATES/COARSE GRID

- Choose a collection of “representative” original system states, and associate each one of them with an aggregate state. Then “interpolate”



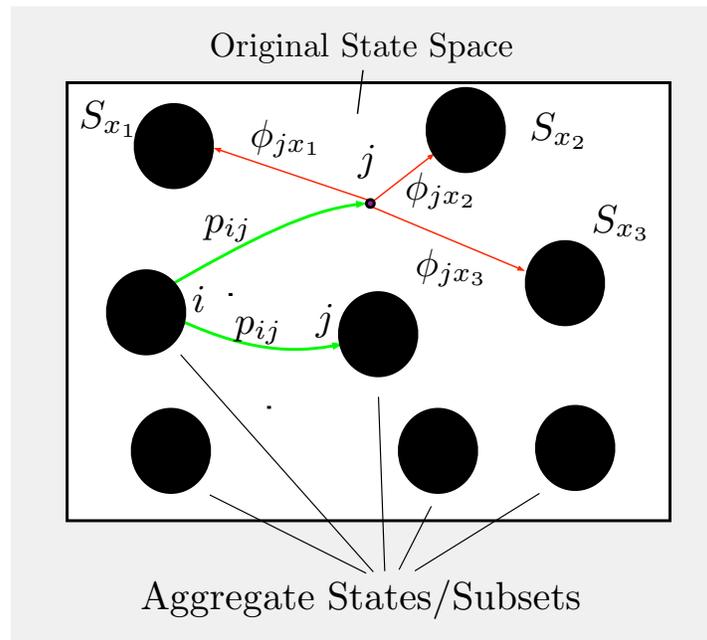
- Disaggregation probabilities are $d_{xi} = 1$ if i is equal to representative state x .
- Aggregation probabilities associate original system states with convex combinations of representative states

$$j \sim \sum_{y \in \mathcal{A}} \phi_{jy} y$$

- Well-suited for Euclidean space discretization
- Extends nicely to continuous state space, including belief space of POMDP

EXAMPLE IV: REPRESENTATIVE FEATURES

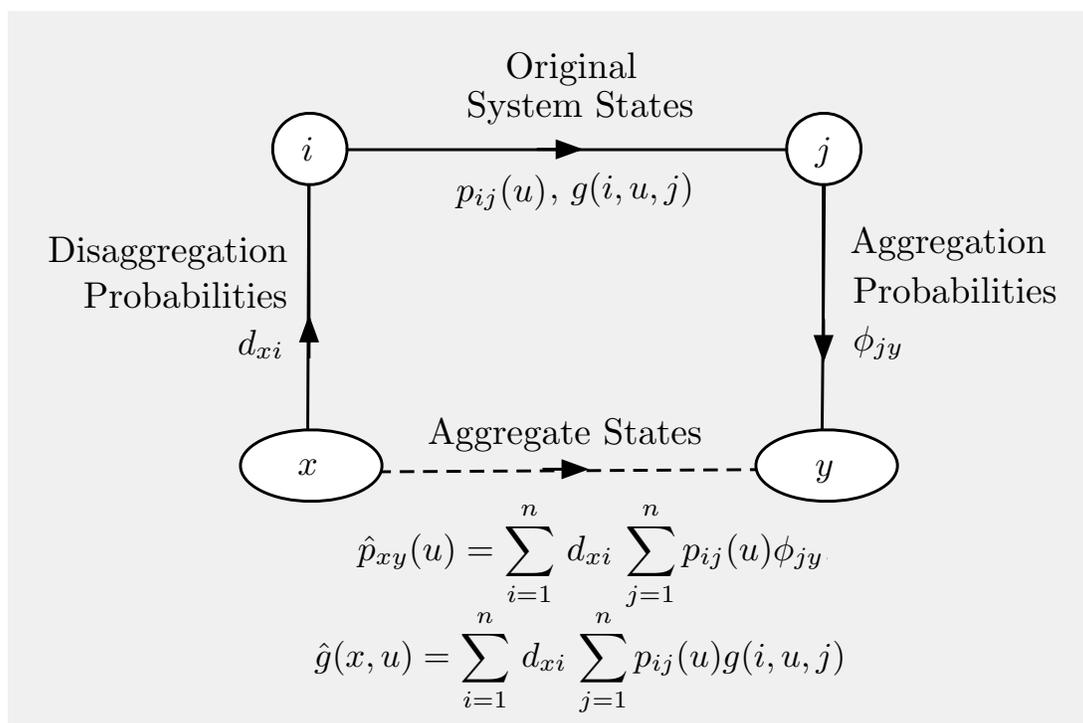
- Choose a collection of “representative” subsets of original system states, and associate each one of them with an aggregate state



- Common case: S_x is a group of states with “similar features”
- Hard aggregation is special case: $\cup_x S_x = \{1, \dots, n\}$
- Aggregation with representative states is special case: S_x consists of just one state
- With rep. features, aggregation approach is a special case of projected equation approach with “seminorm” or “oblique” projection (see text). So the TD methods and multistage Bellman Eq. methodology apply

APPROXIMATE PI BY AGGREGATION

- Consider approximate PI for the original problem, with evaluation done using the aggregate problem (other possibilities exist - see the text)
- **Evaluation of policy μ :** $\tilde{J} = \Phi R$, where $R = DT_\mu(\Phi R)$ (R is the vector of costs of aggregate states corresponding to μ). May use simulation.
- Similar form to the projected equation $\Phi R = \Pi T_\mu(\Phi R)$ (ΦD in place of Π).
- **Advantages:** It has no problem with exploration or with oscillations.
- **Disadvantage:** The rows of D and Φ must be probability distributions.



Q-LEARNING I

- Q-learning has two motivations:
 - Dealing with multiple policies simultaneously
 - Using a model-free approach [no need to know $p_{ij}(u)$, only be able to simulate them]
- The Q-factors are defined by

$$Q^*(i, u) = \sum_{j=1}^n p_{ij}(u) (g(i, u, j) + \alpha J^*(j)), \quad \forall (i, u)$$

- Since $J^* = TJ^*$, we have $J^*(i) = \min_{u \in U(i)} Q^*(i, u)$ so the Q factors solve the equation

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

- $Q^*(i, u)$ can be shown to be the unique solution of this equation. **Reason:** This is Bellman's equation for a system whose states are the original states $1, \dots, n$, together with all the pairs (i, u) .

- **Value iteration:** For all (i, u)

$$Q(i, u) := \sum_{j=1}^n p_{ij}(u) \left(g(i, u, j) + \alpha \min_{u' \in U(j)} Q(j, u') \right)$$

Q-LEARNING II

- Use any probabilistic mechanism to select sequence of pairs (i_k, u_k) [all pairs (i, u) are chosen infinitely often], and for each k , select j_k according to $p_{i_k j}(u_k)$.

- **Q-learning algorithm:** updates $Q(i_k, u_k)$ by

$$Q(i_k, u_k) := (1 - \gamma_k(i_k, u_k))Q(i_k, u_k) + \gamma_k(i_k, u_k) \left(g(i_k, u_k, j_k) + \alpha \min_{u' \in U(j_k)} Q(j_k, u') \right)$$

- Stepsize $\gamma_k(i_k, u_k)$ must converge to 0 at proper rate (e.g., like $1/k$).

- **Important mathematical point:** In the Q-factor version of Bellman's equation the order of expectation and minimization is reversed relative to the ordinary cost version of Bellman's equation:

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

- Q-learning can be shown to converge to true/exact Q-factors (sophisticated stoch. approximation proof).

- **Major drawback:** The large number of pairs (i, u) - no function approximation is used.

Q-FACTOR APPROXIMATIONS

- Basis function approximation for Q -factors:

$$\tilde{Q}(i, u, r) = \phi(i, u)'r$$

- We can use approximate policy iteration and LSPE/LSTD/TD for policy evaluation (exploration issue is acute).

- Optimistic policy iteration methods are frequently used on a heuristic basis.

- **Example** (very optimistic). At iteration k , given r_k and state/control (i_k, u_k) :

(1) Simulate next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$.

(2) Generate control u_{k+1} from

$$u_{k+1} = \arg \min_{u \in U(i_{k+1})} \tilde{Q}(i_{k+1}, u, r_k)$$

(3) Update the parameter vector via

$$r_{k+1} = r_k - (\text{LSPE or TD-like correction})$$

- **Unclear validity**. Solid basis for an important but very special case: optimal stopping (see the text)

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

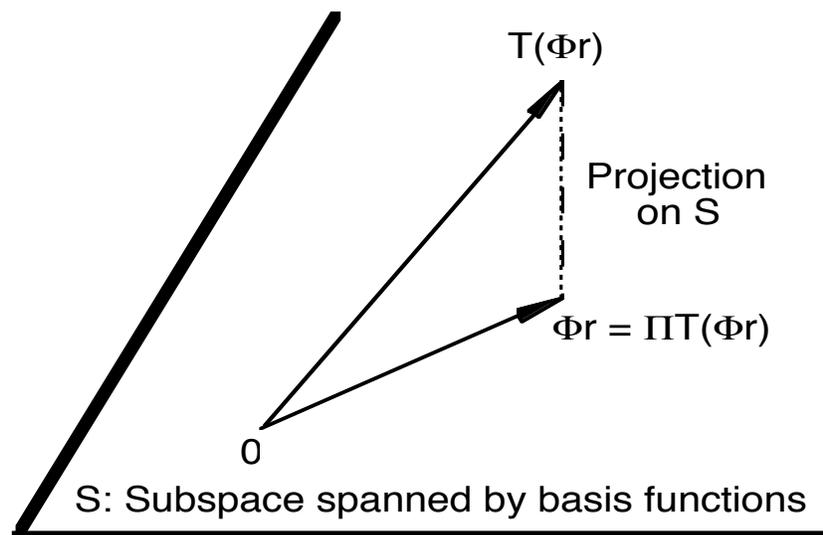
- Additional topics in ADP
- Stochastic shortest path problems
- Average cost problems
- Generalizations
- Basis function adaptation
- Gradient-based approximation in policy space

REVIEW: PROJECTED BELLMAN EQUATION

- **Policy Evaluation:** Approximate Bellman's equation $J = TJ$ is approximated by the **projected equation**

$$\Phi r = \Pi T(\Phi r)$$

which can be solved by a simulation-based methods such as LSPE(λ), LSTD(λ), or TD(λ). (A related approach is **aggregation** - simpler in various ways.)



Indirect method: Solving a projected form of Bellman's equation

- These ideas apply to other (linear) Bellman equations, e.g., for SSP and average cost.
- **Important Issue:** Construct simulation framework where ΠT [or $\Pi T^{(\lambda)}$] is a contraction.

STOCHASTIC SHORTEST PATHS

- Introduce approximation subspace

$$S = \{\Phi r \mid r \in \mathbb{R}^s\}$$

and for a given proper policy, Bellman's equation and its projected version

$$J = TJ = g + PJ, \quad \Phi r = \Pi T(\Phi r)$$

Also its λ -version

$$\Phi r = \Pi T^{(\lambda)}(\Phi r), \quad T^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

- Question: **What should be the norm of projection?** How to implement it by simulation?
- **Speculation based on discounted case:** It should be a weighted Euclidean norm with weight vector $\xi = (\xi_1, \dots, \xi_n)$, where ξ_i should be some type of long-term occupancy probability of state i (which can be generated by simulation).
- But what does “long-term occupancy probability of a state” mean in the SSP context?
- How do we generate infinite length trajectories given that termination occurs with prob. 1?

SIMULATION FOR SSP

- We envision simulation of trajectories up to termination, followed by **restart at state i with some fixed probabilities $q_0(i) > 0$** .

- Then the “long-term occupancy probability of a state” of i is proportional to

$$q(i) = \sum_{t=0}^{\infty} q_t(i), \quad i = 1, \dots, n,$$

where

$$q_t(i) = P(i_t = i), \quad i = 1, \dots, n, \quad t = 0, 1, \dots$$

- We use the projection norm

$$\|J\|_q = \sqrt{\sum_{i=1}^n q(i) (J(i))^2}$$

[Note that $0 < q(i) < \infty$, but q is not a prob. distribution.]

- We can show that $\Pi T^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_q$ (see the next slide).
- LSTD(λ), LSPE(λ), and TD(λ) are possible.

CONTRACTION PROPERTY FOR SSP

- We have $q = \sum_{t=0}^{\infty} q_t$ so

$$q'P = \sum_{t=0}^{\infty} q'_t P = \sum_{t=1}^{\infty} q'_t = q' - q'_0$$

or

$$\sum_{i=1}^n q(i) p_{ij} = q(j) - q_0(j), \quad \forall j$$

- To verify that PT is a contraction, we show that there exists $\beta < 1$ such that $\|Pz\|_q^2 \leq \beta \|z\|_q^2$ for all $z \in \mathfrak{R}^n$.
- For all $z \in \mathfrak{R}^n$, we have

$$\begin{aligned} \|Pz\|_q^2 &= \sum_{i=1}^n q(i) \left(\sum_{j=1}^n p_{ij} z_j \right)^2 \leq \sum_{i=1}^n q(i) \sum_{j=1}^n p_{ij} z_j^2 \\ &= \sum_{j=1}^n z_j^2 \sum_{i=1}^n q(i) p_{ij} = \sum_{j=1}^n (q(j) - q_0(j)) z_j^2 \\ &= \|z\|_q^2 - \|z\|_{q_0}^2 \leq \beta \|z\|_q^2 \end{aligned}$$

where

$$\beta = 1 - \min_j \frac{q_0(j)}{q(j)}$$

AVERAGE COST PROBLEMS

- Consider a single policy to be evaluated, with single recurrent class, no transient states, and steady-state probability vector $\xi = (\xi_1, \dots, \xi_n)$.
- The average cost, denoted by η , is

$$\eta = \lim_{N \rightarrow \infty} \frac{1}{N} E \left\{ \sum_{k=0}^{N-1} g(x_k, x_{k+1}) \mid x_0 = i \right\}, \quad \forall i$$

- Bellman's equation is $J = FJ$ with

$$FJ = g - \eta e + PJ$$

where e is the unit vector $e = (1, \dots, 1)$.

- The projected equation and its λ -version are

$$\Phi r = \Pi F(\Phi r), \quad \Phi r = \Pi F^{(\lambda)}(\Phi r)$$

- A problem here is that F is not a contraction with respect to any norm (since $e = Pe$).
- However, $\Pi F^{(\lambda)}$ turns out to be a contraction with respect to $\|\cdot\|_\xi$ assuming that e does not belong to S and $\lambda > 0$ [the case $\lambda = 0$ is exceptional, but can be handled - see the text]. LSTD(λ), LSPE(λ), and TD(λ) are possible.

GENERALIZATION/UNIFICATION

- Consider approx. solution of $x = T(x)$, where

$$T(x) = Ax + b, \quad A \text{ is } n \times n, \quad b \in \mathbb{R}^n$$

by solving the projected equation $y = \Pi T(y)$, where Π is projection on a subspace of basis functions (with respect to some Euclidean norm).

- We will generalize from DP to the case where **A is arbitrary**, subject only to

$$I - \Pi A : \text{invertible}$$

There are ways to deal with the case where $I - \Pi A$ is singular (iterative methods, see the text).

- Benefits of generalization:
 - Unification/higher perspective for projected equation (and aggregation) methods in approximate DP
 - An extension to a broad new area of applications, based on an approx. DP perspective
- Challenge: Dealing with less structure
 - Lack of contraction
 - Absence of a Markov chain

GENERALIZED PROJECTED EQUATION

- Let Π be projection with respect to

$$\|x\|_{\xi} = \sqrt{\sum_{i=1}^n \xi_i x_i^2}$$

where $\xi \in \mathbb{R}^n$ is a probability distribution with positive components.

- If r^* is the solution of the projected equation, we have $\Phi r^* = \Pi(A\Phi r^* + b)$ or

$$r^* = \arg \min_{r \in \mathbb{R}^s} \sum_{i=1}^n \xi_i \left(\phi(i)'r - \sum_{j=1}^n a_{ij} \phi(j)'r^* - b_i \right)^2$$

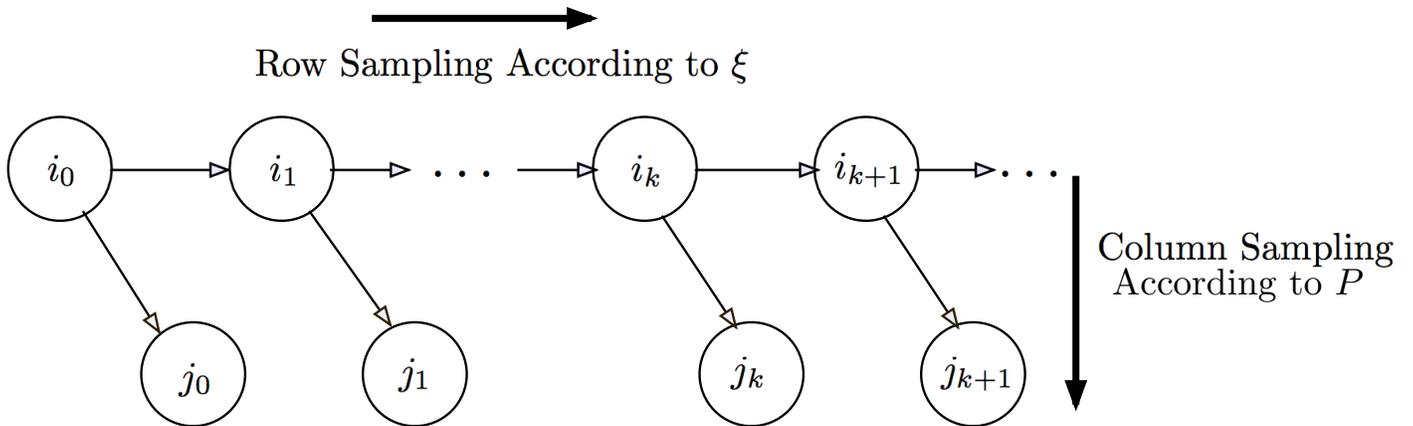
where $\phi(i)'$ denotes the i th row of the matrix Φ .

- Optimality condition/equivalent form:

$$\sum_{i=1}^n \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^n a_{ij} \phi(j) \right)' r^* = \sum_{i=1}^n \xi_i \phi(i) b_i$$

- The two expected values can be approximated by simulation

SIMULATION MECHANISM



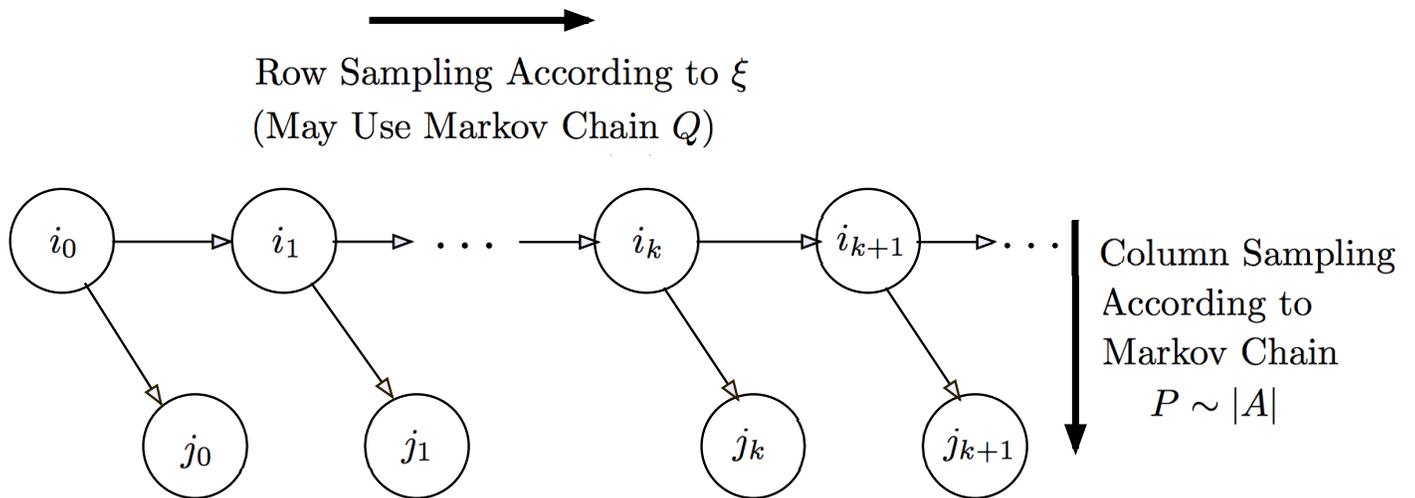
- **Row sampling:** Generate sequence $\{i_0, i_1, \dots\}$ according to ξ , i.e., relative frequency of each row i is ξ_i
- **Column sampling:** Generate $\{(i_0, j_0), (i_1, j_1), \dots\}$ according to some transition probability matrix P with

$$p_{ij} > 0 \quad \text{if} \quad a_{ij} \neq 0,$$

i.e., for each i , the relative frequency of (i, j) is p_{ij} (connection to **importance sampling**)

- Row sampling **may** be done using a Markov chain with transition matrix Q (**unrelated to P**)
- Row sampling **may also be done without** a Markov chain - just sample rows according to some known distribution ξ (e.g., a uniform)

ROW AND COLUMN SAMPLING



- Row sampling \sim State Sequence Generation in DP. Affects:
 - The projection norm.
 - Whether ΠA is a contraction.
- Column sampling \sim Transition Sequence Generation in DP.
 - Can be totally unrelated to row sampling. Affects the sampling/simulation error.
 - “Matching” P with $|A|$ is beneficial (has an effect like in importance sampling).
- Independent row and column sampling allows **exploration at will!** Resolves the exploration problem that is critical in approximate policy iteration.

LSTD-LIKE METHOD

- Optimality condition/equivalent form of projected equation

$$\sum_{i=1}^n \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^n a_{ij} \phi(j) \right)' r^* = \sum_{i=1}^n \xi_i \phi(i) b_i$$

- The two expected values are approximated by row and column sampling (batch $0 \rightarrow t$).
- We solve the linear equation

$$\sum_{k=0}^t \phi(i_k) \left(\phi(i_k) - \frac{a_{i_k j_k}}{p_{i_k j_k}} \phi(j_k) \right)' r_t = \sum_{k=0}^t \phi(i_k) b_{i_k}$$

- We have $r_t \rightarrow r^*$, **regardless of ΠA being a contraction** (by law of large numbers; see next slide).
- Issues of singularity or near-singularity of $I - \Pi A$ may be important; see the text.
- An LSPE-like method is also possible, but requires that ΠA is a contraction.
- Under the assumption $\sum_{j=1}^n |a_{ij}| \leq 1$ for all i , there are conditions that guarantee contraction of ΠA ; see the text.

JUSTIFICATION W/ LAW OF LARGE NUMBERS

- We will match terms in the exact optimality condition and the simulation-based version.
- Let $\hat{\xi}_i^t$ be the relative frequency of i in row sampling up to time t .
- We have

$$\frac{1}{t+1} \sum_{k=0}^t \phi(i_k) \phi(i_k)' = \sum_{i=1}^n \hat{\xi}_i^t \phi(i) \phi(i)' \approx \sum_{i=1}^n \xi_i \phi(i) \phi(i)'$$

$$\frac{1}{t+1} \sum_{k=0}^t \phi(i_k) b_{i_k} = \sum_{i=1}^n \hat{\xi}_i^t \phi(i) b_i \approx \sum_{i=1}^n \xi_i \phi(i) b_i$$

- Let \hat{p}_{ij}^t be the relative frequency of (i, j) in column sampling up to time t .

$$\begin{aligned} \frac{1}{t+1} \sum_{k=0}^t \frac{a_{i_k j_k}}{p_{i_k j_k}} \phi(i_k) \phi(j_k)' \\ &= \sum_{i=1}^n \hat{\xi}_i^t \sum_{j=1}^n \hat{p}_{ij}^t \frac{a_{ij}}{p_{ij}} \phi(i) \phi(j)' \\ &\approx \sum_{i=1}^n \xi_i \sum_{j=1}^n a_{ij} \phi(i) \phi(j)' \end{aligned}$$

BASIS FUNCTION ADAPTATION I

- An important issue in ADP is how to select basis functions.
- A possible approach is to introduce basis functions that are parametrized by a vector θ , and optimize over θ , i.e., solve the problem

$$\min_{\theta \in \Theta} F(\tilde{J}(\theta))$$

where $\tilde{J}(\theta)$ is the solution of the projected equation.

- One example is

$$F(\tilde{J}(\theta)) = \|\tilde{J}(\theta) - T(\tilde{J}(\theta))\|^2$$

- Another example is

$$F(\tilde{J}(\theta)) = \sum_{i \in I} |J(i) - \tilde{J}(\theta)(i)|^2,$$

where I is a subset of states, and $J(i)$, $i \in I$, are the costs of the policy at these states calculated directly by simulation.

BASIS FUNCTION ADAPTATION II

- Some algorithm may be used to minimize $F(\tilde{J}(\theta))$ over θ .
- A challenge here is that the algorithm should use low-dimensional calculations.
- One possibility is to use a form of random search (the cross-entropy method); see the paper by Menache, Mannor, and Shimkin (Annals of Oper. Res., Vol. 134, 2005)
- Another possibility is to use a gradient method. For this it is necessary to estimate the partial derivatives of $\tilde{J}(\theta)$ with respect to the components of θ .
- It turns out that by differentiating the projected equation, these partial derivatives can be calculated using low-dimensional operations. See the references in the text.

APPROXIMATION IN POLICY SPACE I

- Consider an average cost problem, where the problem data are parametrized by a vector r , i.e., a cost vector $g(r)$, transition probability matrix $P(r)$. Let $\eta(r)$ be the (scalar) average cost per stage, satisfying Bellman's equation

$$\eta(r)e + h(r) = g(r) + P(r)h(r)$$

where $h(r)$ is the differential cost vector.

- Consider minimizing $\eta(r)$ over r (here the data dependence on control is encoded in the parametrization). Other than random search, we can try to solve the problem by **nonlinear programming/gradient descent** methods.

- **Important fact:** If $\Delta\eta$, Δg , ΔP are the changes in η, g, P due to a small change Δr from a given r , we have

$$\Delta\eta = \xi'(\Delta g + \Delta P h),$$

where ξ is the steady-state probability distribution/vector corresponding to $P(r)$, and all the quantities above are evaluated at r .

APPROXIMATION IN POLICY SPACE II

- **Proof of the gradient formula:** We have, by “differentiating” Bellman’s equation,

$$\Delta\eta(r)\cdot e + \Delta h(r) = \Delta g(r) + \Delta P(r)h(r) + P(r)\Delta h(r)$$

By left-multiplying with ξ' ,

$$\xi' \Delta\eta(r)\cdot e + \xi' \Delta h(r) = \xi' (\Delta g(r) + \Delta P(r)h(r)) + \xi' P(r)\Delta h(r)$$

Since $\xi' \Delta\eta(r) \cdot e = \Delta\eta(r)$ and $\xi' = \xi' P(r)$, this equation simplifies to

$$\Delta\eta = \xi'(\Delta g + \Delta P h)$$

- Since we don’t know ξ , we cannot implement a gradient-like method for minimizing $\eta(r)$. An alternative is to use “sampled gradients”, i.e., generate a simulation trajectory (i_0, i_1, \dots) , and change r once in a while, in the direction of a simulation-based estimate of $\xi'(\Delta g + \Delta P h)$.
- **Important Fact:** $\Delta\eta$ can be viewed as an expected value!
- There is much research on this subject, see the text.

6.231 DYNAMIC PROGRAMMING

OVERVIEW-EPILOGUE

LECTURE OUTLINE

- Finite horizon problems
 - Deterministic vs Stochastic
 - Perfect vs Imperfect State Info
- Infinite horizon problems
 - Stochastic shortest path problems
 - Discounted problems
 - Average cost problems

FINITE HORIZON PROBLEMS - ANALYSIS

- Perfect state info
 - A general formulation - Basic problem, DP algorithm
 - A few nice problems admit analytical solution
- Imperfect state info
 - Reduction to perfect state info - Sufficient statistics
 - Very few nice problems that admit analytical solution
 - Finite-state problems admit reformulation as perfect state info problems whose states are prob. distributions (the belief vectors)

FINITE HORIZON PROBS - EXACT COMP. SOL.

- Deterministic finite-state problems
 - Equivalent to shortest path
 - A wealth of fast algorithms
 - Hard combinatorial problems are a special case (but # of states grows exponentially)
- Stochastic perfect state info problems
 - The DP algorithm is the only choice
 - Curse of dimensionality is big bottleneck
- Imperfect state info problems
 - Forget it!
 - Only trivial examples admit an exact computational solution

FINITE HORIZON PROBS - APPROX. SOL.

- Many techniques (and combinations thereof) to choose from
- Simplification approaches
 - Certainty equivalence
 - Problem simplification
 - Rolling horizon
 - Aggregation - Coarse grid discretization
- Limited lookahead combined with:
 - Rollout
 - MPC (an important special case)
 - Feature-based cost function approximation
- Approximation in policy space
 - Gradient methods
 - Random search

INFINITE HORIZON PROBLEMS - ANALYSIS

- A more extensive theory
- Bellman's equation
- Optimality conditions
- Contraction mappings
- A few nice problems admit analytical solution
- Idiosyncracies of problems with no underlying contraction
- Idiosyncracies of average cost problems
- Elegant analysis

INF. HORIZON PROBS - EXACT COMP. SOL.

- Value iteration
 - Variations (Gauss-Seidel, asynchronous, etc)
- Policy iteration
 - Variations (asynchronous, based on value iteration, optimistic, etc)
- Linear programming
- Elegant algorithmic analysis
- Curse of dimensionality is major bottleneck

INFINITE HORIZON PROBS - ADP

- Approximation in value space (over a subspace of basis functions)
- Approximate policy evaluation
 - Direct methods
 - Indirect methods (projected equation methods)
 - Aggregation methods - Coarse grid discretization
- Q-Learning
 - Exact Q-factor computation by simulation
 - Approximate Q-factor computation by simulation
 - Projected equation methods for optimal stopping
 - Aggregation-based Q-learning
- Approximate LP
- Approximation in policy space
 - Gradient methods
 - Random search