

LECTURE SLIDES ON DYNAMIC PROGRAMMING
BASED ON LECTURES GIVEN AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASS
FALL 2001

DIMITRI P. BERTSEKAS

**These lecture slides are based on the book:
“Dynamic Programming and Optimal Control:
2nd edition,” Vols. I and II, Athena Scientific,
2001, by Dimitri P. Bertsekas; see**

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

**The slides are copyrighted but may be freely
reproduced and distributed for any noncom-
mercial purpose.**

6.231 DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

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

BASIC STRUCTURE OF STOCHASTIC DP

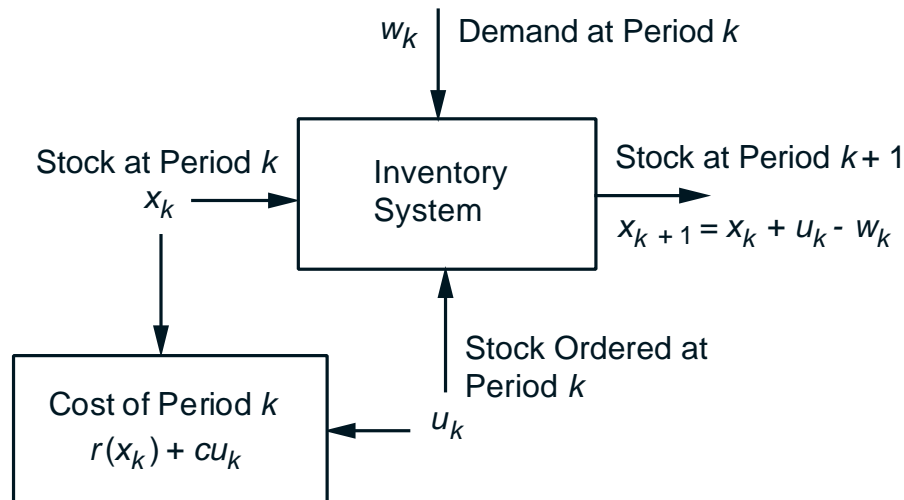
- Discrete-time system

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

- 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

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

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

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

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

- 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.,

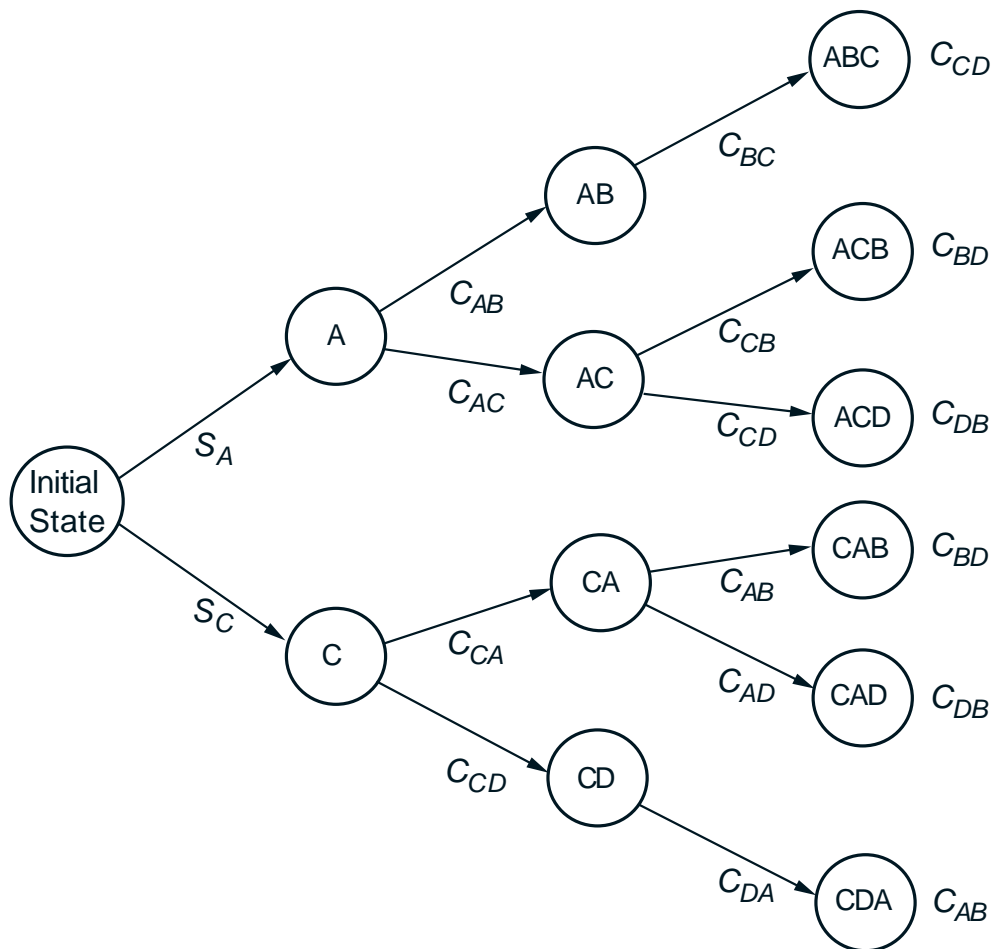
$$u_k \in U(x_k)$$

- w_k is random and generated according to a distribution

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

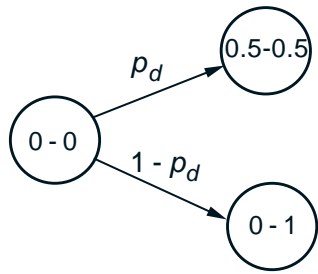
DETERMINISTIC FINITE-STATE PROBLEMS

- 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

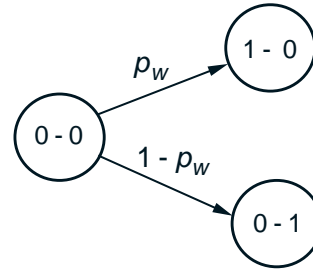


STOCHASTIC FINITE-STATE PROBLEMS

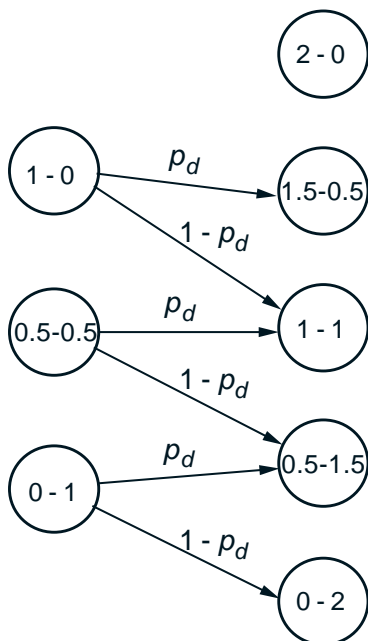
- Example: Find two-game chess match strategy
- *Timid* play draws with prob. $p_d > 0$ and loses with prob. $1 - p_d$. *Bold* play wins with prob. $p_w < 1/2$ and loses with prob. $1 - p_w$



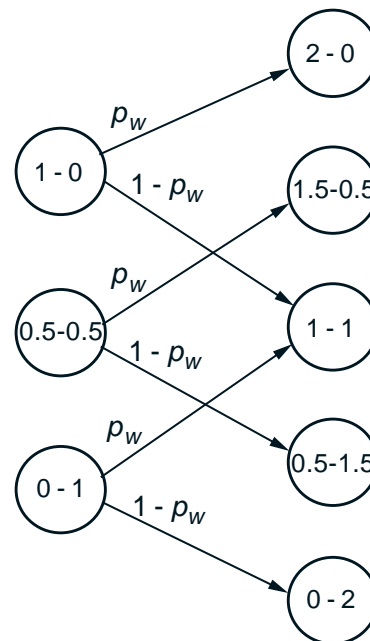
1st Game / Timid Play



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 $u_k \in U(x_k)$
- 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 policy π^* :

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

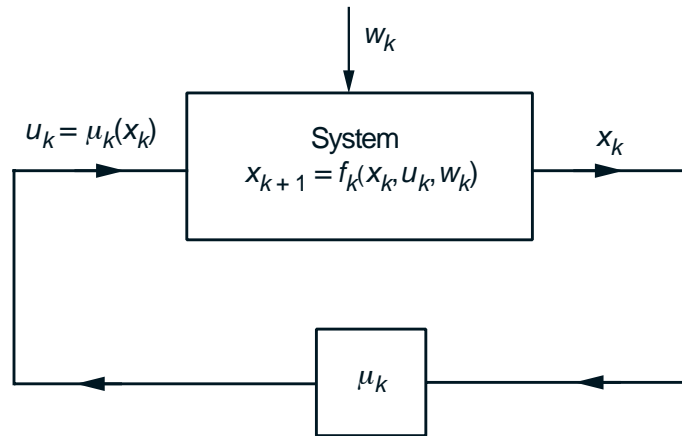
When produced by DP, it is independent of x_0 .

- Optimal cost function

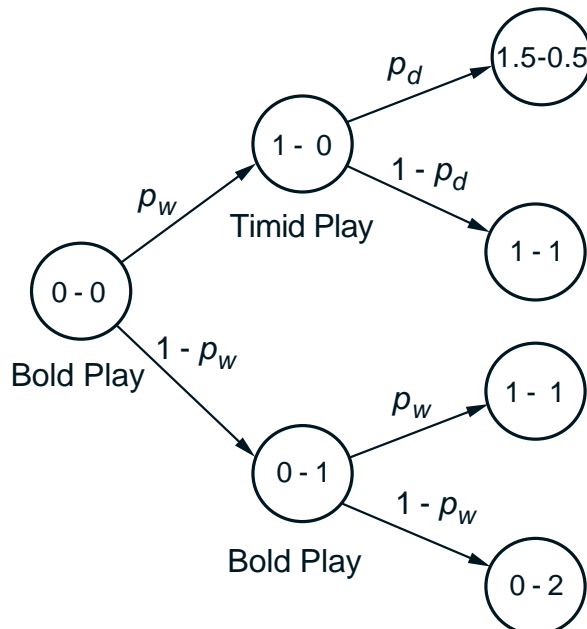
$$J^*(x_0) = \min_{\pi} J_\pi(x_0)$$

SIGNIFICANCE OF FEEDBACK

- Open-loop versus closed-loop policies



- In deterministic problems open loop is a good as closed loop
- Chess match example; value of information



6.231 DYNAMIC PROGRAMMING

LECTURE 2

LECTURE OUTLINE

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

BASIC PROBLEM

- System $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \dots, N - 1$
- Control constraints $u_k \in U(x_k)$
- 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 policy π^* :

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

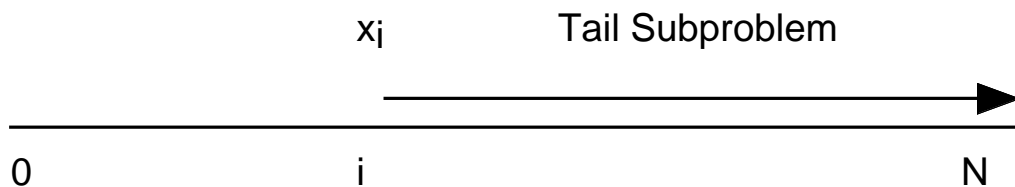
- Optimal cost function

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

PRINCIPLE OF OPTIMALITY

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$ be an 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

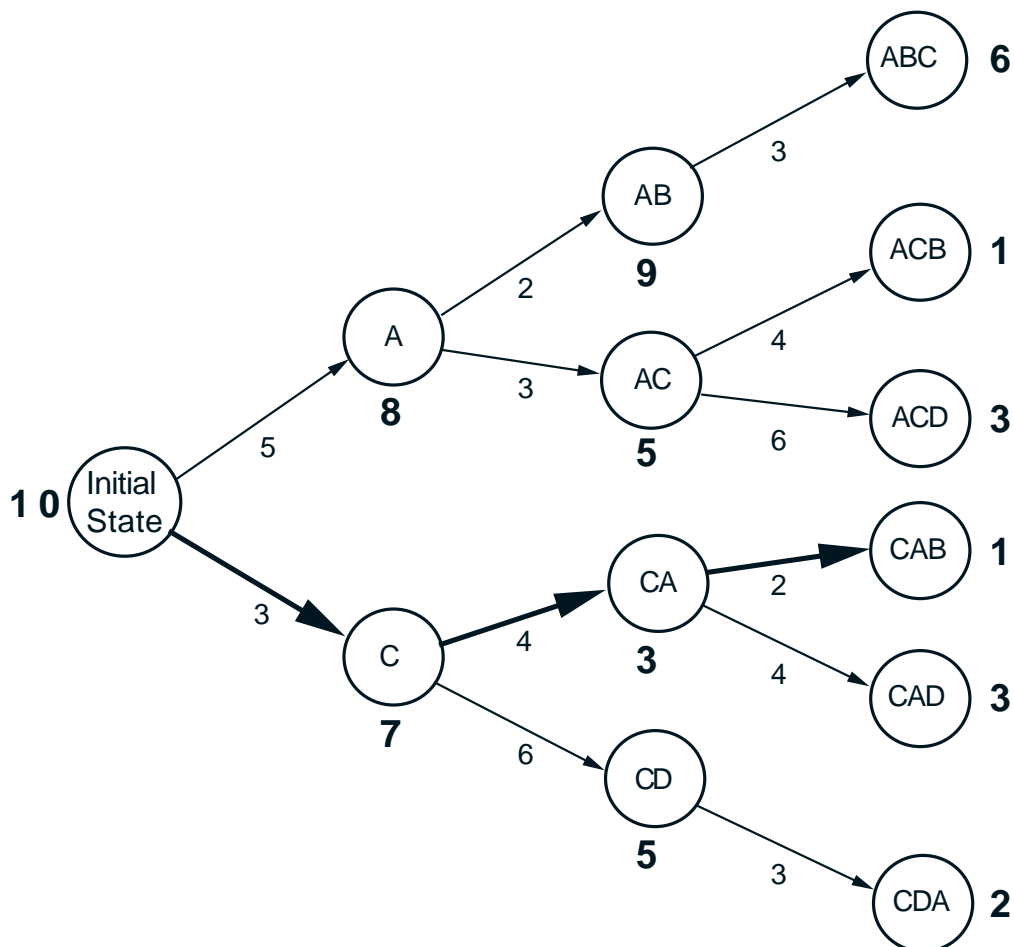
$$\mathbf{E} \left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$



- Then the truncated policy $\{\mu_i^*, \mu_{i+1}^*, \dots, \mu_{N-1}^*\}$ is optimal for this subproblem (assuming x_i is a “relevant state” at time i).
- 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.

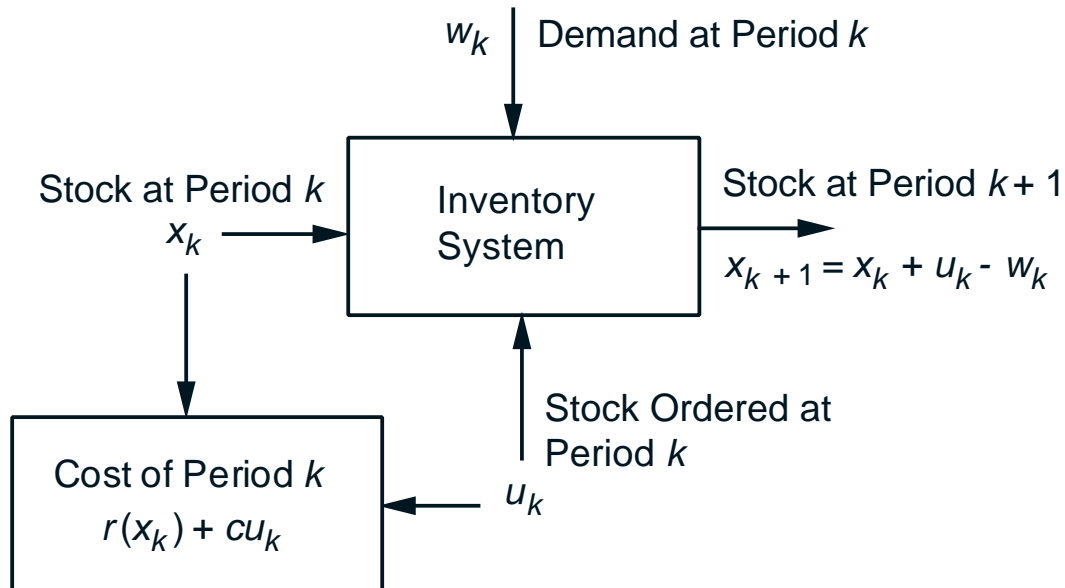
DETERMINISTIC SCHEDULING EXAMPLE

- 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:

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

- Tail Subproblems of Length $N - k$:

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

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 of the algorithm, is equal to the optimal cost $J^*(x_0)$. Furthermore, if $u_k^* = \mu_k^*(x_k)$ minimizes in the right side above for each x_k and k , the policy

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

is optimal.

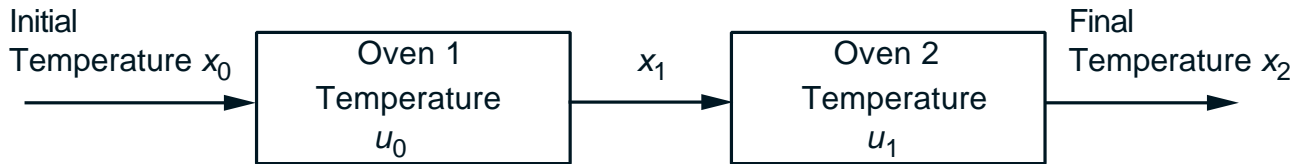
- Justification: Proof by induction that $J_k(x_k)$ is equal to $J_k^*(x_k)$, defined as the optimal cost of the tail subproblem that starts at time k at state x_k .
- Note that ALL the tail subproblems are solved in addition to the original problem, and the intensive computational requirements.

PROOF OF THE INDUCTION STEP

- 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 the state.
- 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:

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

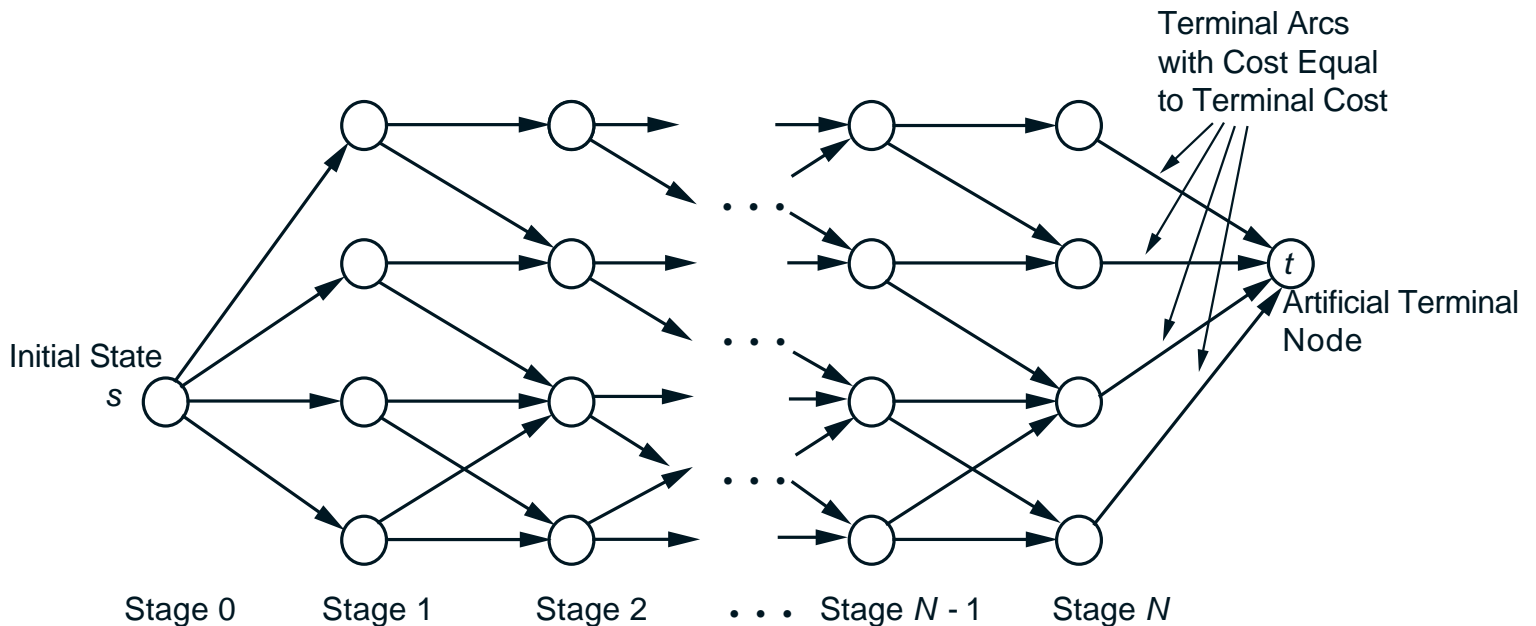
6.231 DYNAMIC PROGRAMMING

LECTURE 3

LECTURE OUTLINE

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

BASIC DETERMINISTIC PROBLEM



- States \Leftrightarrow Nodes
- Controls \Leftrightarrow Arcs
- Control sequences (open-loop) \Leftrightarrow 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
- a_{it}^N : Terminal cost of state $i \in S_N$
- Cost of control sequence \Leftrightarrow Cost/length of the corresponding path

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:

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

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

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* .

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 $a_{ii} = 0$.

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

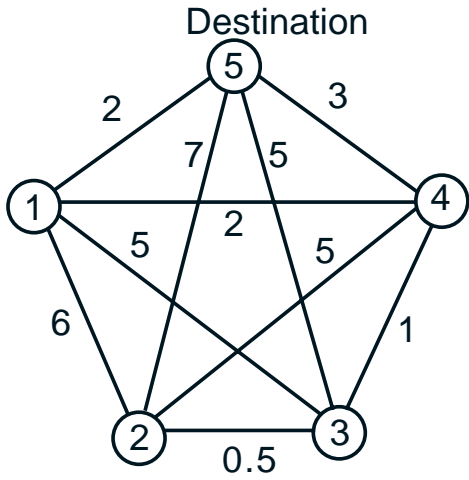
$J_0(i)$: Cost of the optimal path from i to t .

- DP algorithm:

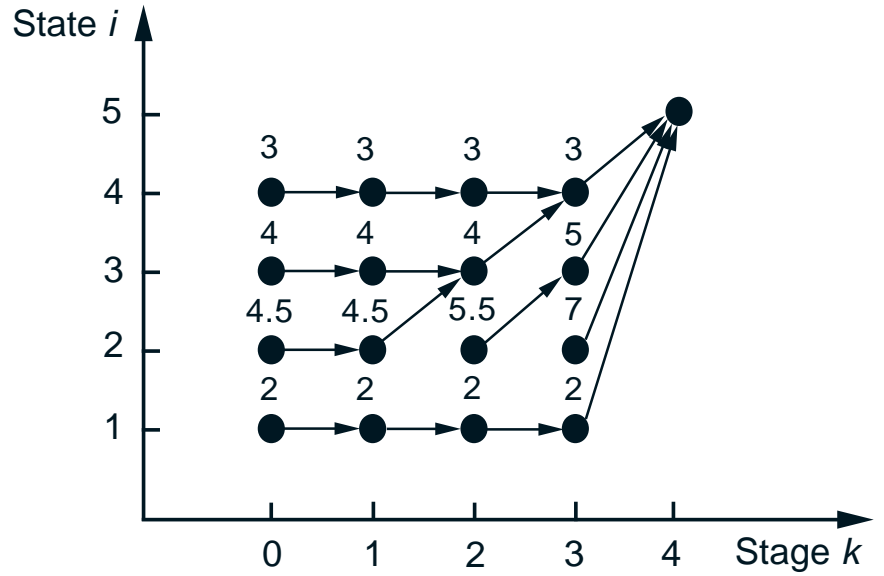
$$J_k(i) = \min_{j=1, \dots, N} [a_{ij} + J_{k+1}(j)], \quad k = 0, 1, \dots, N-2,$$

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

EXAMPLE



(a)



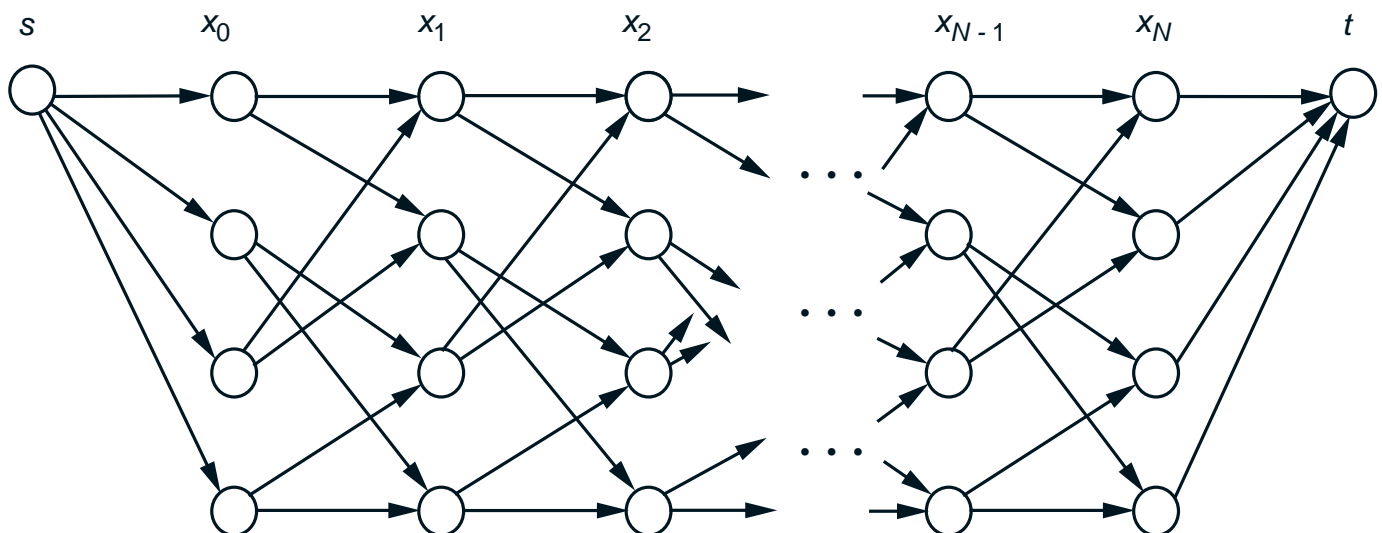
(b)

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

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

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

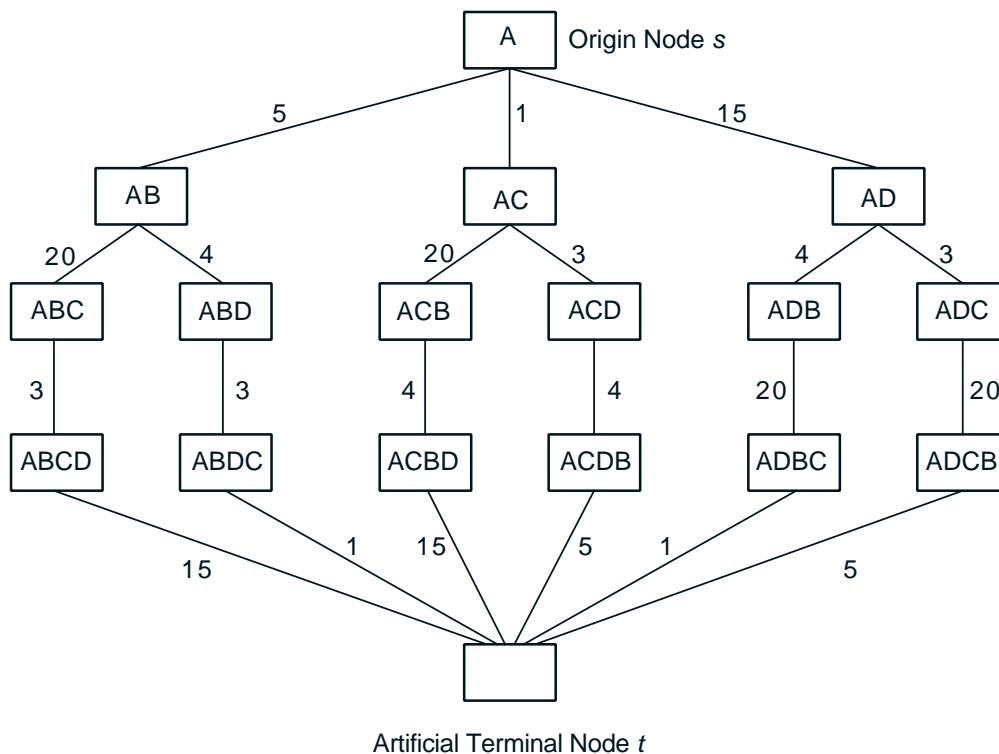
$$\text{minimize } -\ln(\pi_{x_0}) - \sum_{k=1}^N \ln(p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k))$$

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

- Origin s , destination t , lengths a_{ij} are ≥ 0 .
- Idea is to progressively discover shorter paths from the origin s to every other node i
- d_i (label of i): Length of the shortest path found
- UPPER: 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

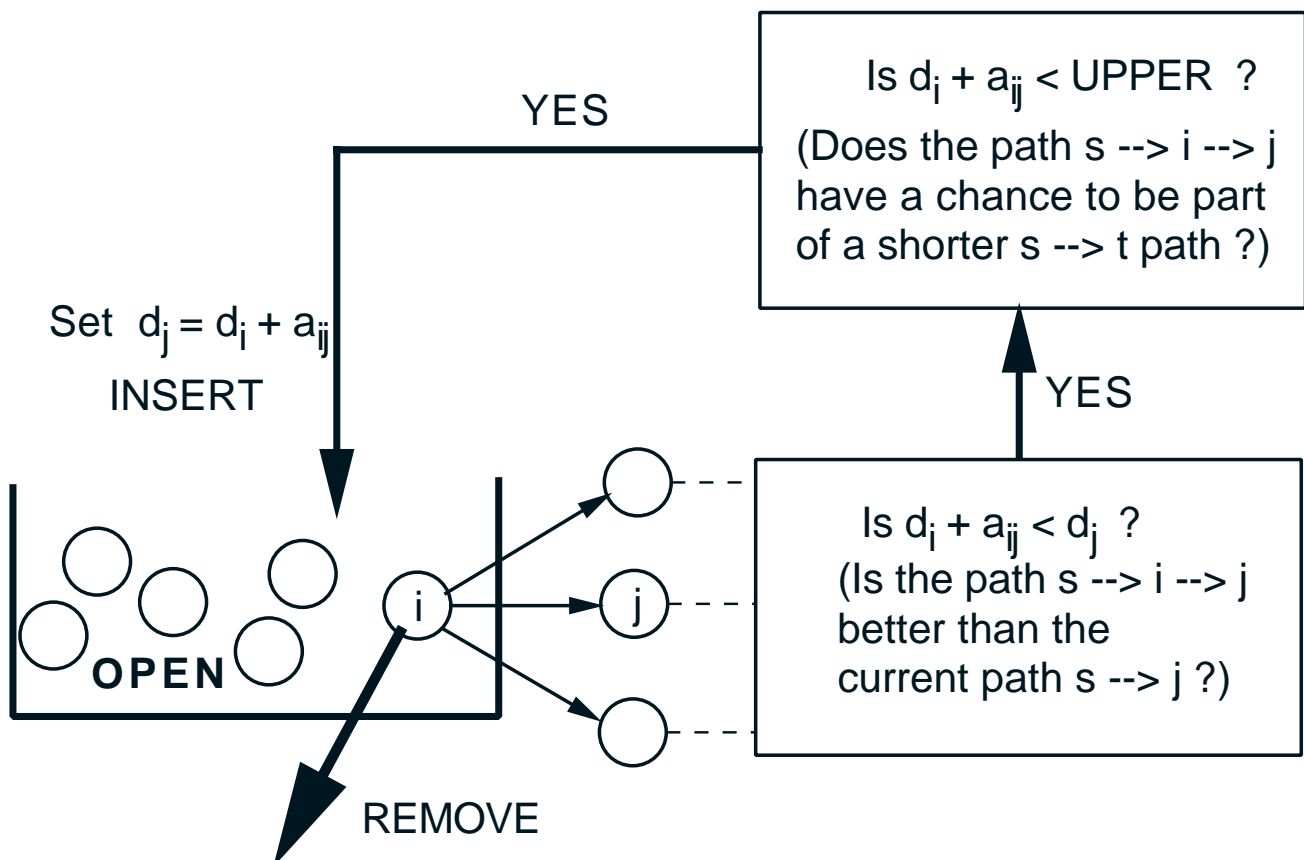
Step 1: Remove a node i from OPEN and for each child j of i , execute step 2.

Step 2: 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 .

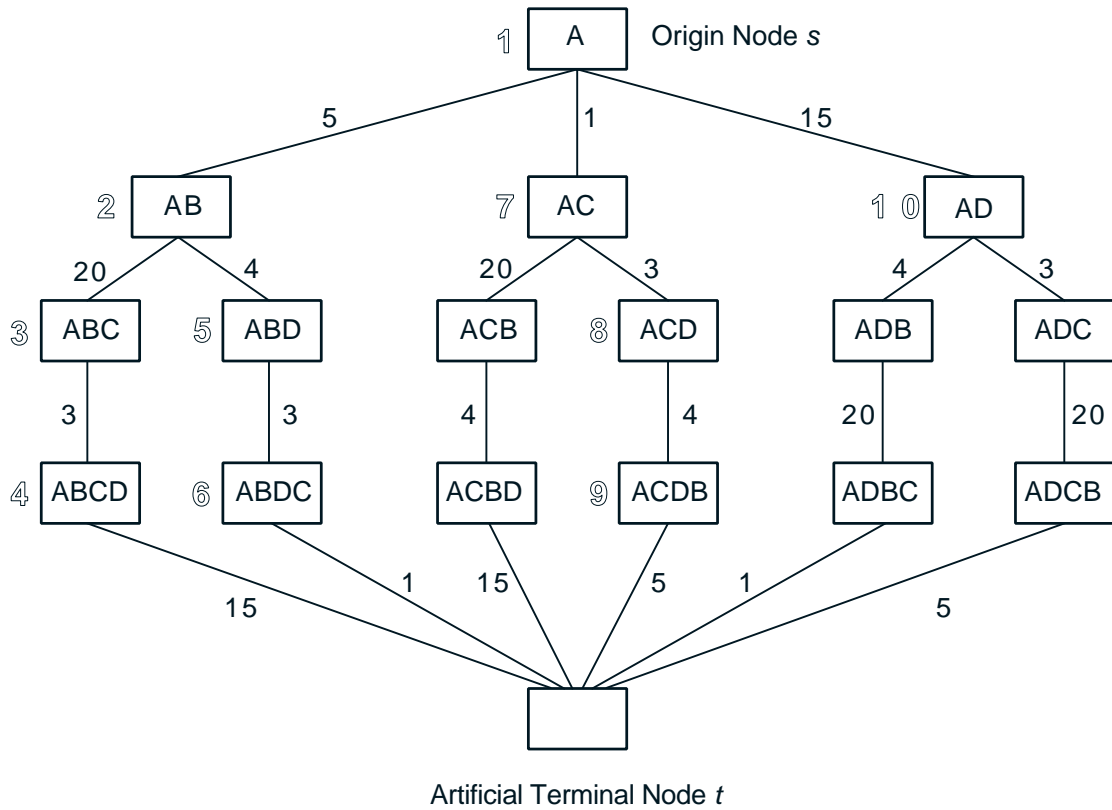
Step 3: If OPEN is empty, terminate; else go to step 1.

VISUALIZATION/EXPLANATION

- Origin s , destination t , lengths a_{ij} that are ≥ 0 .
- d_i (label of i): Length of the shortest path found thus far (initially $d_i = \infty$ except $d_s = 0$). The label d_i is implicitly associated with an $s \rightarrow i$ path.
- UPPER: 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

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 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

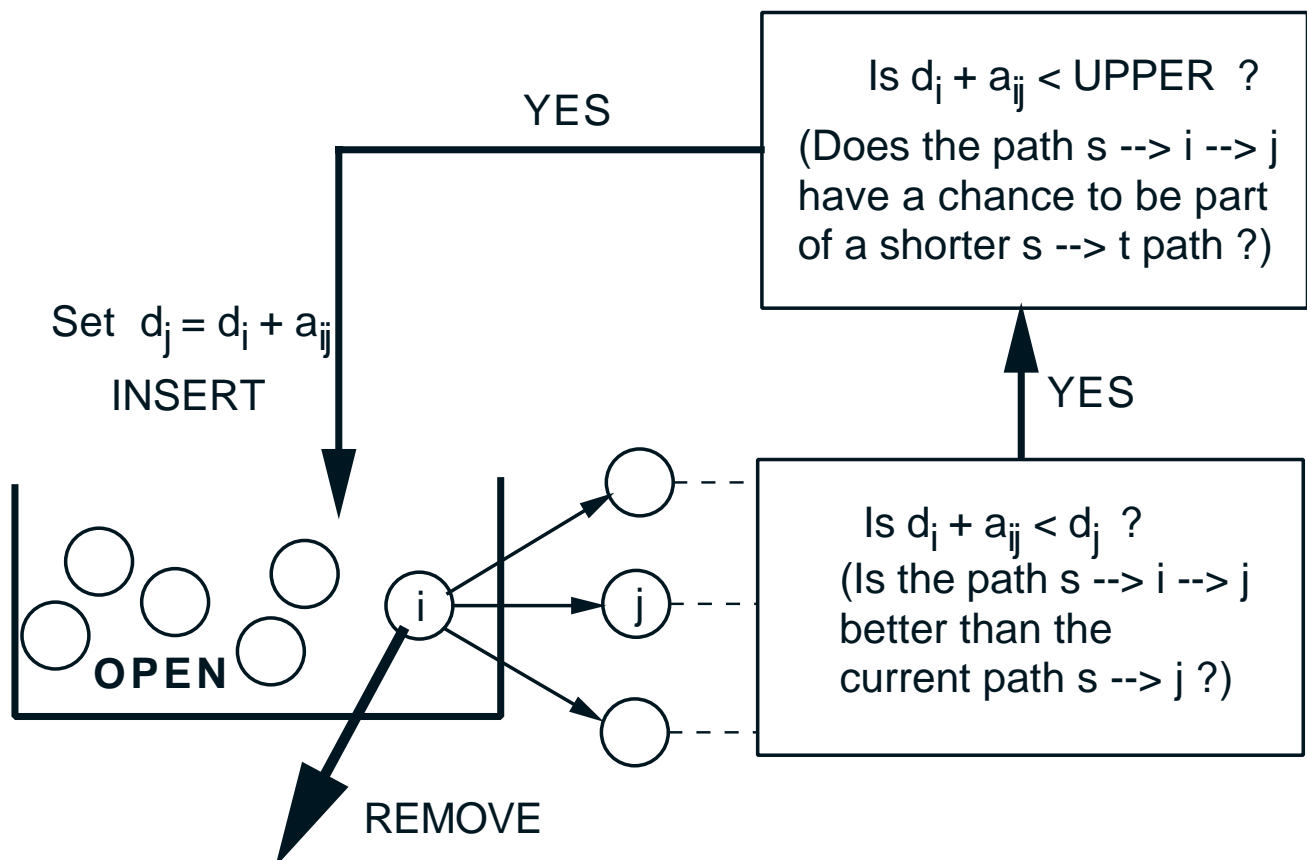
LECTURE 4

LECTURE OUTLINE

- Label correcting methods for shortest paths
- Variants of label correcting methods
- Branch-and-bound as a shortest path algorithm
- Linear-quadratic problems

LABEL CORRECTING METHODS

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

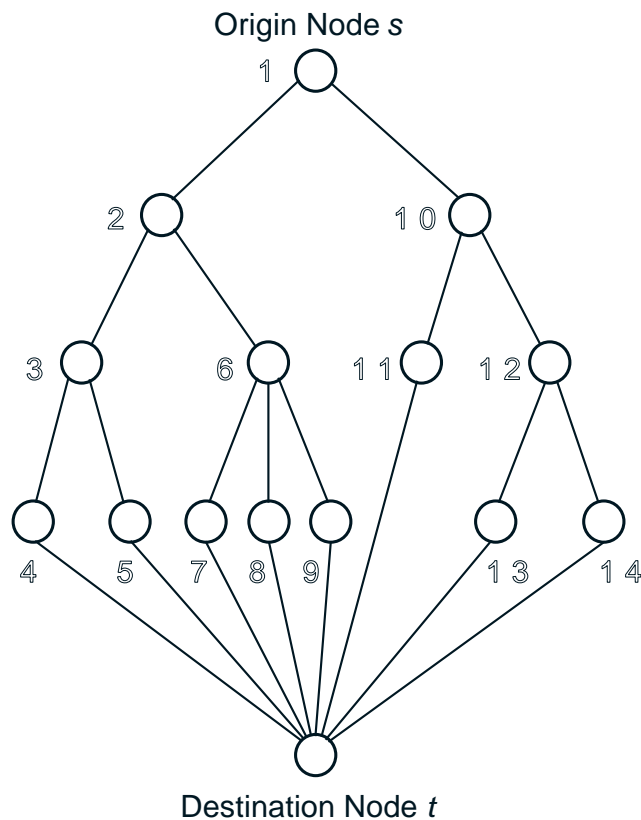


MAKING THE METHOD EFFICIENT

- Reduce the value of UPPER as quickly as possible
 - Try to discover “good” $s \rightarrow t$ paths early in the course of the algorithm
- Keep the number of reentries into OPEN low
 - Try to remove from OPEN nodes with small label first
 - For an acyclic graph each node will enter OPEN at most once, so this issue does not arise
- Reduce the overhead for selecting the node to be removed from OPEN
- These objectives are often in conflict. They give rise to a large variety of distinct implementations.

NODE SELECTION METHODS

- Depth-first search: Remove from the top of OPEN and insert at the top of OPEN
 - Has low memory storage properties (OPEN is not too long). Reduces UPPER quickly.



- Best-first search (Dijkstra): Remove from OPEN a node with minimum value of label.
 - Interesting property: Each node will be inserted in OPEN at most once.
 - Many implementations/approximations

ADVANCED INITIALIZATION

- Instead of starting from $d_i = \infty$ for all $i \neq s$, start with

$d_i =$ length of some path from s to i (or $d_i = \infty$)

$$\text{OPEN} = \{i \neq t \mid d_i < \infty\}$$

- Motivation: Get a small starting value of UPPER.
- No node with shortest distance \leq initial value of UPPER will enter OPEN
- Good practical idea:
 - Run a heuristic (or use common sense) to get a “good” starting path P from s to t
 - Use as UPPER the length of P , and as d_i the path distances of all nodes i along P
- Very useful also in reoptimization, where we solve the same problem with slightly different data

VARIANTS OF LABEL CORRECTING METHODS

- If a lower bound h_j of the true shortest distance from j to t is known, use the test

$$d_i + a_{ij} + h_j < \text{UPPER}$$

for entry into OPEN, instead of

$$d_i + a_{ij} < \text{UPPER}$$

- If an upper bound m_j of the true shortest distance from j to t is known, then if $d_j + m_j < \text{UPPER}$, reduce UPPER to $d_j + m_j$
- Important use: Branch-and-bound algorithm for discrete optimization

BRANCH-AND-BOUND METHOD

- Problem: Minimize $f(x)$ over a *finite* set of feasible solutions X .
- Idea of branch-and-bound: Partition the feasible set into smaller subsets, and then calculate certain bounds on the attainable cost within some of the subsets to eliminate from further consideration other subsets.

Bounding Principle

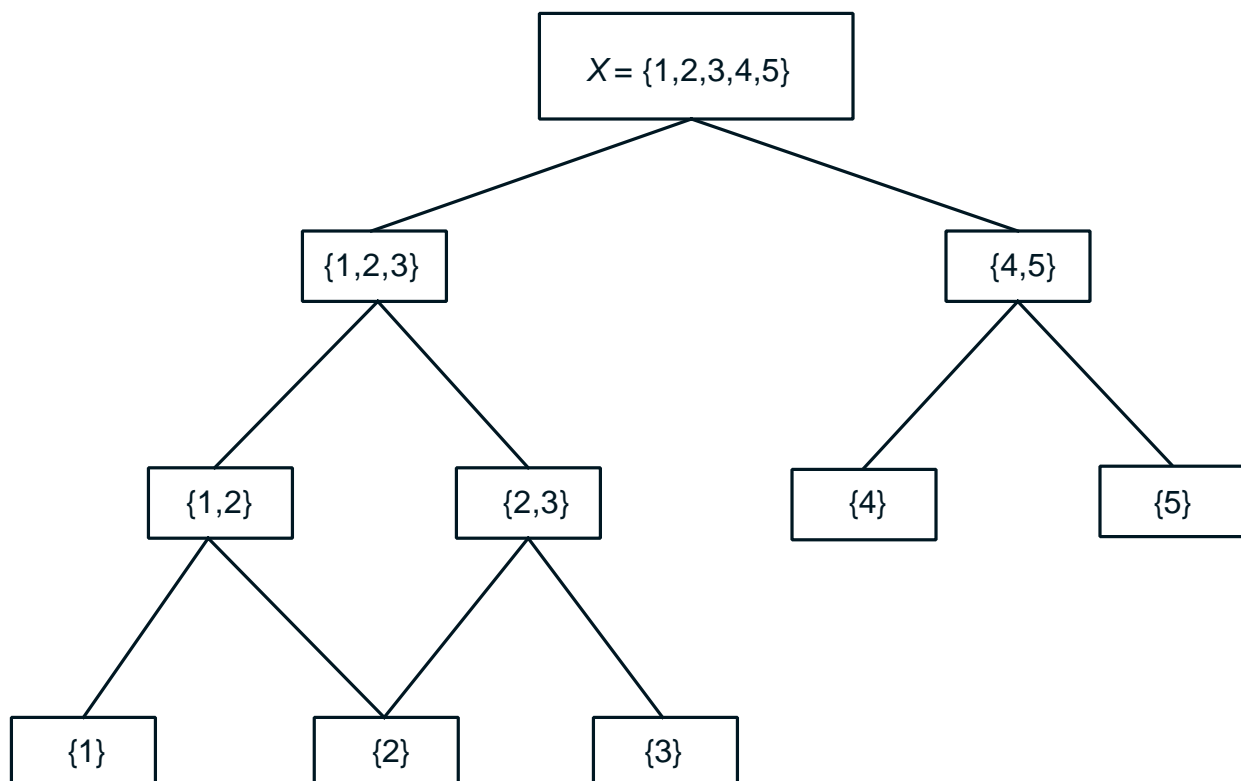
Given two subsets $Y_1 \subset X$ and $Y_2 \subset X$, suppose that we have bounds

$$\underline{f}_1 \leq \min_{x \in Y_1} f(x), \quad \bar{f}_2 \geq \min_{x \in Y_2} f(x).$$

Then, if $\bar{f}_2 \leq \underline{f}_1$, the solutions in Y_1 may be disregarded since their cost cannot be smaller than the cost of the best solution in Y_2 .

SHORTEST PATH IMPLEMENTATION

- Acyclic graph/partition of X into subsets. The leafs consist of single solutions.
- Upper/Lower bounds \underline{f}_Y and \bar{f}_Y for the minimum cost over each subset Y can be calculated
- Arc (Y, Z) has length $\underline{f}_Z - \underline{f}_Y$
- Shortest distance from X to $Y = \underline{f}_Y - \underline{f}_X$
- Shortest path to the set of leafs gives the optimal cost and optimal solution



BRANCH-AND-BOUND ALGORITHM

Step 1: Remove a node Y from OPEN. For each child Y_j of Y , do the following: If $\underline{f}_{Y_j} < \text{UPPER}$, then place Y_j in OPEN. If in addition $\bar{f}_{Y_j} < \text{UPPER}$, then set $\text{UPPER} = \bar{f}_{Y_j}$, and if Y_j consists of a single solution, mark that solution as being the best solution found so far.

Step 2: (Termination Test) If OPEN is nonempty, go to step 1. Otherwise, terminate; the best solution found so far is optimal.

- It is neither practical nor necessary to generate a priori the acyclic graph (we generate it as we go).
- Keys to branch-and-bound:
 - Generate as sharp as possible upper and lower bounds at each node
 - Have a good partitioning and node selection strategy
- Method involves a lot of art, may be prohibitively time-consuming, but is guaranteed to find an optimal solution.

LINEAR-QUADRATIC PROBLEMS

- System: $x_{k+1} = A_k x_k + B_k u_k + w_k, k = 0, \dots, N - 1$

- Quadratic cost

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

- 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:

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

DERIVATION

- By induction and straightforward calculation, verify that

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

where the matrices L_k are given by

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

and where the symmetric positive semidefinite matrices K_k are given by the algorithm

$$K_N = Q_N,$$

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

- 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 $E\{w_k\} = 0$).

6.231 DYNAMIC PROGRAMMING

LECTURE 5

LECTURE OUTLINE

- Linear quadratic problem
- 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 and straightforward calculation, verify that

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

where the matrices L_k are given by

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

and where the symmetric positive semidefinite matrices K_k are given by the algorithm

$$K_N = Q_N,$$

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

- 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 EQUATION

- 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 the unique positive semidefinite 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

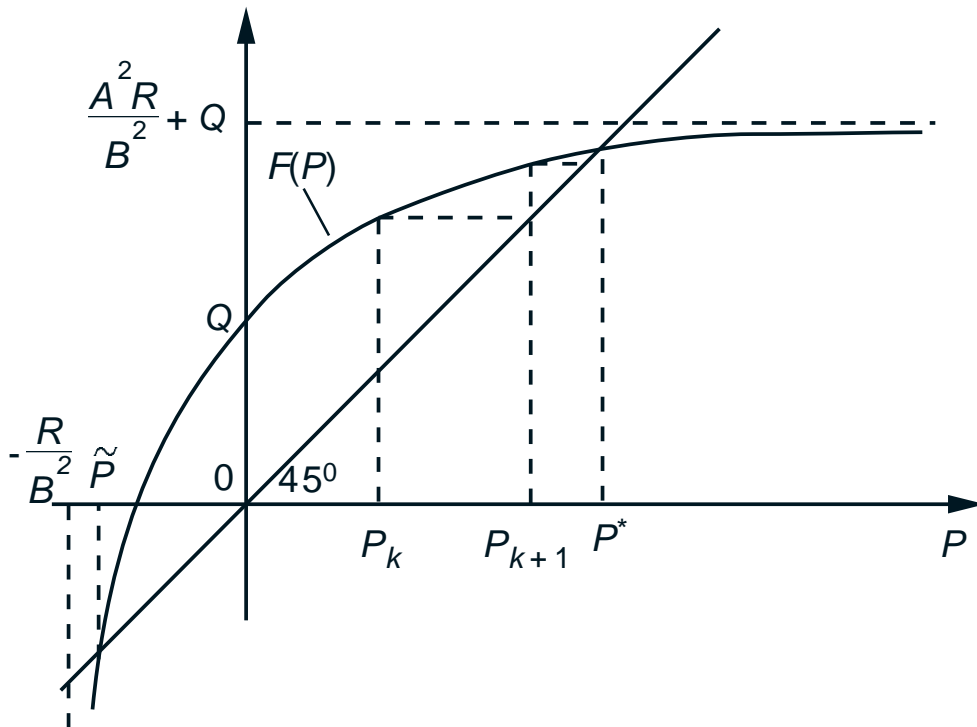
$$L = -(B'KB + R)^{-1}B'KA,$$

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

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

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

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

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

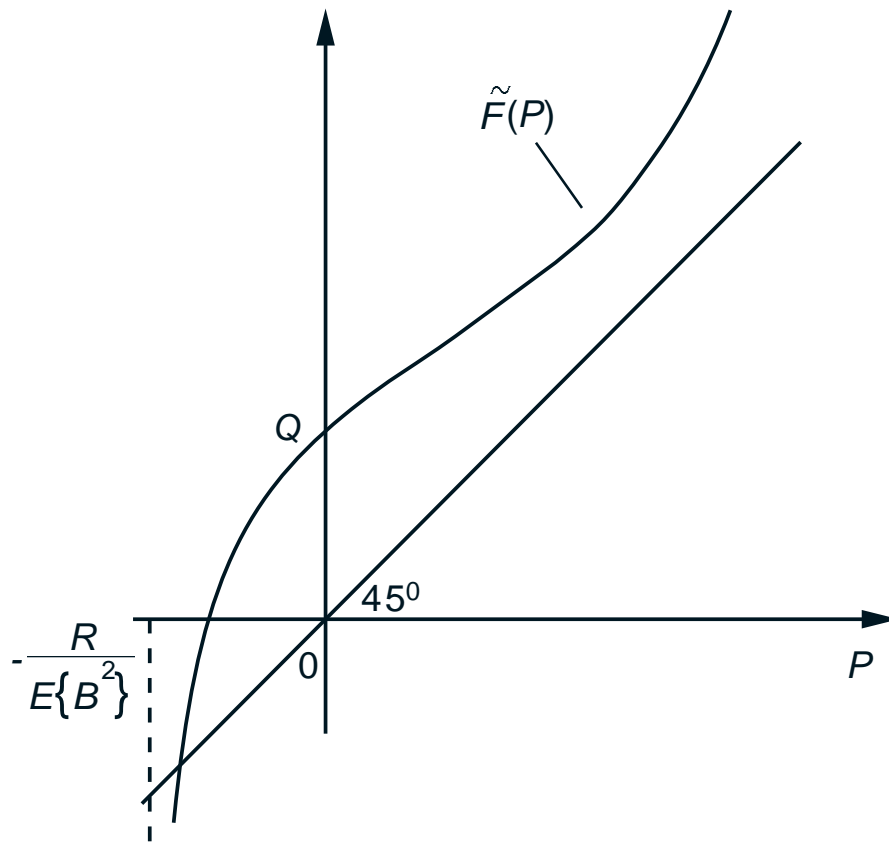
and where the matrices K_k are given by

$$K_N = Q_N,$$

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

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

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

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

INVENTORY CONTROL

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

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

- Minimize

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

where, for some $p > 0$ and $h > 0$,

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

- DP algorithm:

$$J_N(x_N) = 0,$$

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

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

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

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

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

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

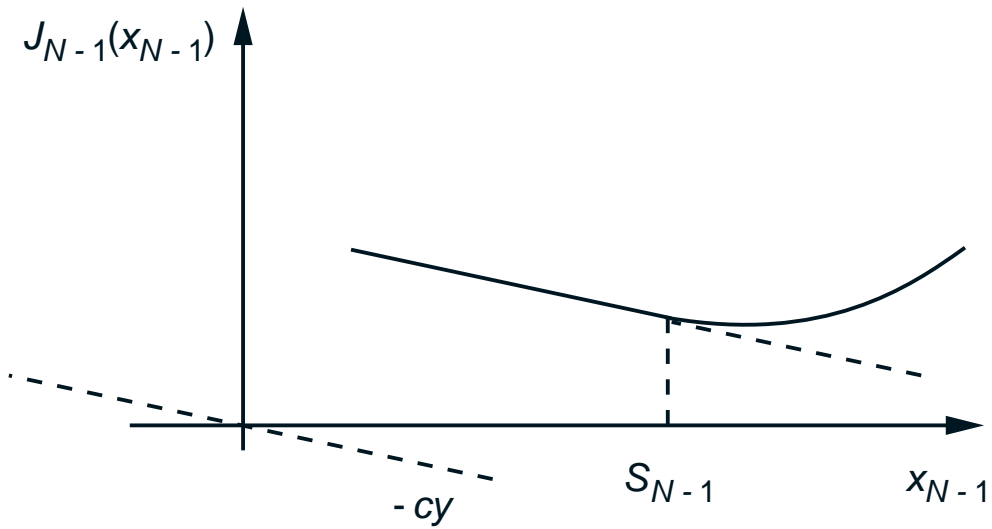
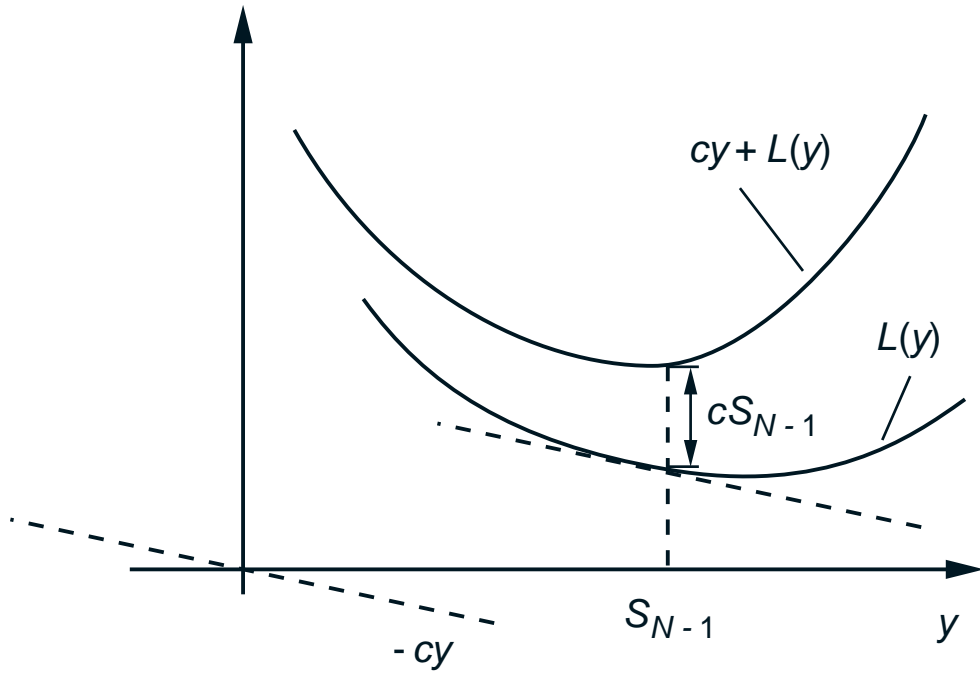
where S_k minimizes $G_k(y)$.

- 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

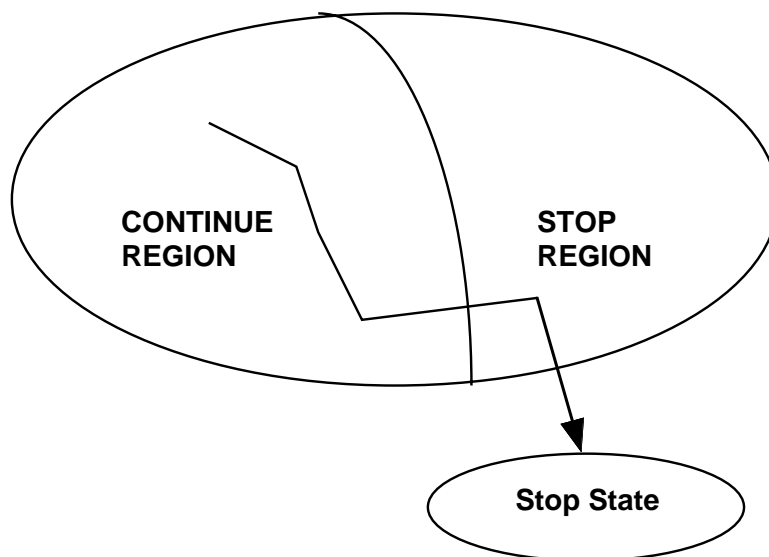
LECTURE 6

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):

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

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

- Optimal policy;

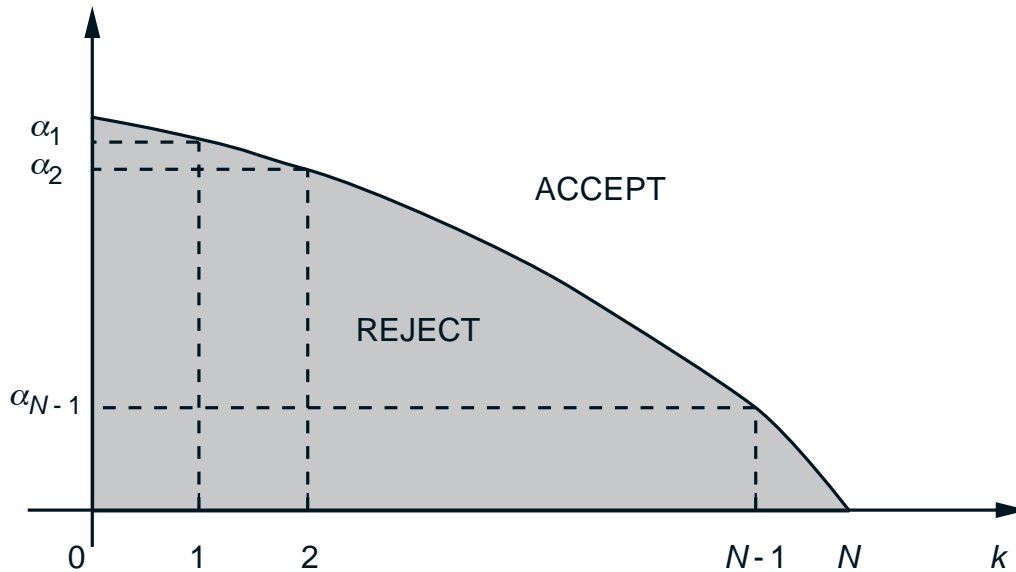
accept the offer x_k if $x_k > \alpha_k$,

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

where

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

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

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

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

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

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

- 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

- 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 “index”, and it is optimal to select the task that has the maximum 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)

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

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

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

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, $\mathbf{E} \{ \text{reward of } L \} \geq \mathbf{E} \{ \text{reward of } L' \}$, 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 $k = 0, 1, \dots, N-1$,

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

6.231 DYNAMIC PROGRAMMING

LECTURE 7

LECTURE OUTLINE

- Deterministic continuous-time optimal control
- Examples
- Connection with the calculus of variations
- The Hamilton-Jacobi-Bellman equation

PROBLEM FORMULATION

- We have a continuous-time dynamic system

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

where

- $x(t) \in \mathbb{R}^n$ is the state vector at time t
 - $u(t) \in U \subset \mathbb{R}^m$ is the control vector at time t , U is the control constraint set
 - T is the terminal time.
- We assume that, for any admissible control trajectory $\{u(t) \mid t \in [0, T]\}$ (piecewise continuous functions $\{u(t) \mid t \in [0, T]\}$ with $u(t) \in U$ for all $t \in [0, T]$), the system has a unique solution.
 - 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 a cost function of the form

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

- 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

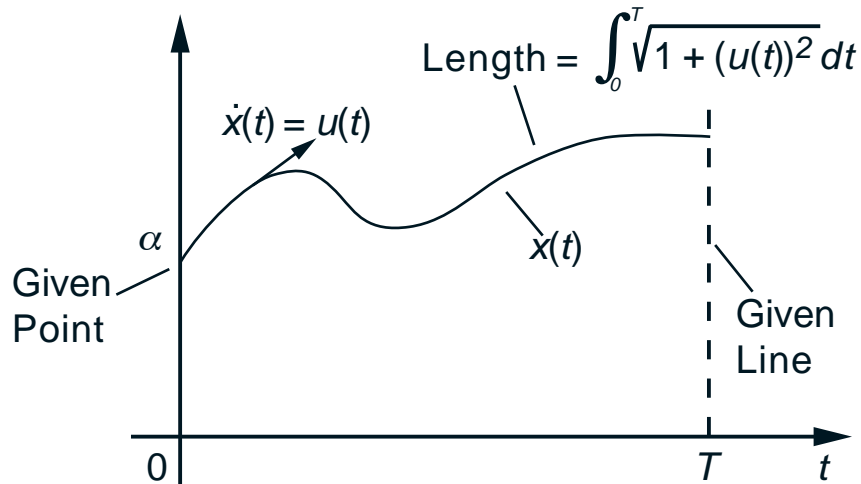
$$\int_0^T (1 - u(t))x(t)dt$$

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

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

$$\text{subject to } x(0) = \alpha.$$

- Reformulation as an optimal control problem:

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

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

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:

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

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 ,

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

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).
- It is 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 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.

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

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

- 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

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

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 8

LECTURE OUTLINE

- Deterministic continuous-time optimal control
- From the HJB equation to the Pontryagin Minimum Principle
- Examples

THE HJB EQUATION

- Continuous-time dynamic system

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

- Cost function

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

- $J^*(t, x)$: optimal cost-to-go from x at time t
- HJB equation: 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 $J^*(T, x) = h(x)$.

- Verification theorem: If we can find a solution, it must be equal to the optimal cost-to-go function. Also a (closed-loop) policy $\mu^*(t, x)$ s.t.

$$\mu^*(t, x) \text{ attains the min for each } (t, x)$$

is optimal.

HJB EQ. ALONG AN OPTIMAL TRAJECTORY

- Observation I: An optimal control-state trajectory pair $\{(u^*(t), x^*(t)) \mid t \in [0, T]\}$ satisfies for all $t \in [0, T]$

$$u^*(t) = \arg \min_{u \in U} [g(x^*(t), u) + \nabla_x J^*(t, x^*(t))' f(x^*(t), u)]. \quad (1)$$

- Observation II: To obtain an optimal control trajectory $\{u^*(t) \mid t \in [0, T]\}$ via this equation, we don't need to know $\nabla_x J^*(t, x)$ for *all* (t, x) - only the time function

$$p(t) = \nabla_x J^*(t, x^*(t)), \quad t \in [0, T].$$

- It turns out that calculating $p(t)$ is often easier than calculating $J^*(t, x)$ or $\nabla_x J^*(t, x)$ for all (t, x) .
- Pontryagin's minimum principle is just Eq. (1) together with an equation for calculating $p(t)$, called the *adjoint* equation.
- Also, Pontryagin's minimum principle is valid much more generally, even in cases where $J^*(t, x)$ is not differentiable and the HJB has no solution.

DERIVING THE ADJOINT EQUATION

- The HJB equation holds as an identity for all (t, x) , so it can be differentiated [the gradient of the RHS with respect to (t, x) is identically 0].
- We need a tool for differentiation of “minimum” functions.

Lemma: Let $F(t, x, u)$ be a continuously differentiable function of $t \in \mathfrak{R}$, $x \in \mathfrak{R}^n$, and $u \in \mathfrak{R}^m$, and let U be a convex subset of \mathfrak{R}^m . Assume that $\mu^*(t, x)$ is a continuously differentiable function such that

$$\mu^*(t, x) = \arg \min_{u \in U} F(t, x, u), \quad \text{for all } t, x.$$

Then

$$\nabla_t \left\{ \min_{u \in U} F(t, x, u) \right\} = \nabla_t F(t, x, \mu^*(t, x)), \quad \text{for all } t, x,$$

$$\nabla_x \left\{ \min_{u \in U} F(t, x, u) \right\} = \nabla_x F(t, x, \mu^*(t, x)), \quad \text{for all } t, x.$$

DIFFERENTIATING THE HJB EQUATION I

- We set to zero the gradient with respect to x and t of the function

$$g(x, \mu^*(t, x)) + \nabla_t J^*(t, x) + \nabla_x J^*(t, x)' f(x, \mu^*(t, x))$$

and we rely on the Lemma to disregard the terms involving the derivatives of $\mu^*(t, x)$ with respect to t and x .

- We obtain for all (t, x) ,

$$\begin{aligned} 0 &= \nabla_x g(x, \mu^*(t, x)) + \nabla_{xt}^2 J^*(t, x) \\ &+ \nabla_{xx}^2 J^*(t, x) f(x, \mu^*(t, x)) + \nabla_x f(x, \mu^*(t, x)) \nabla_x J^*(t, x) \end{aligned}$$

$$0 = \nabla_{tt}^2 J^*(t, x) + \nabla_{xt}^2 J^*(t, x)' f(x, \mu^*(t, x)),$$

where $\nabla_x f(x, \mu^*(t, x))$ is the matrix

$$\nabla_x f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

DIFFERENTIATING THE HJB EQUATION II

- The preceding equations hold for all (t, x) . We specialize them along an optimal state and control trajectory $\{(x^*(t), u^*(t)) \mid t \in [0, T]\}$, where $u^*(t) = \mu^*(t, x^*(t))$ for all $t \in [0, T]$.

- We have $\dot{x}^*(t) = f(x^*(t), u^*(t))$, so the terms

$$\nabla_{xt}^2 J^*(t, x^*(t)) + \nabla_{xx}^2 J^*(t, x^*(t)) f(x^*(t), u^*(t))$$

$$\nabla_{tt}^2 J^*(t, x^*(t)) + \nabla_{xt}^2 J^*(t, x^*(t))' f(x^*(t), u^*(t))$$

are equal to the total derivatives

$$\frac{d}{dt} (\nabla_x J^*(t, x^*(t))), \quad \frac{d}{dt} (\nabla_t J^*(t, x^*(t))).$$

- Denoting $p(t) = \nabla_x J^*(t, x^*(t))$, and $p_0(t) = \nabla_t J^*(t, x^*(t))$, we obtain

$$\dot{p}(t) = -\nabla_x f(x^*(t), u^*(t)) p(t) - \nabla_x g(x^*(t), u^*(t))$$

and $\dot{p}_0(t) = 0$ or equivalently, $p_0(t) = \text{constant}$, for all $t \in [0, T]$.

PONTRYAGIN MINIMUM PRINCIPLE

- Define the Hamiltonian function

$$H(x, u, p) = g(x, u) + p' f(x, u).$$

- **Minimum Principle:** Let $\{u^*(t) \mid t \in [0, T]\}$ be an optimal control trajectory and let $\{x^*(t) \mid t \in [0, T]\}$ be the corresponding state trajectory. Let also $p(t)$ be the solution of the adjoint equation

$$\dot{p}(t) = -\nabla_x H(x^*(t), u^*(t), p(t)),$$

with the boundary condition

$$p(T) = \nabla h(x^*(T)).$$

Then, for all $t \in [0, T]$,

$$u^*(t) = \arg \min_{u \in U} H(x^*(t), u, p(t)).$$

Furthermore, there is a constant C such that

$$H(x^*(t), u^*(t), p(t)) = C, \quad \text{for all } t \in [0, T].$$

2-POINT BOUNDARY PROBLEM VIEW

- The minimum principle is a necessary condition for optimality and can be used to identify candidates for optimality.
- We need to solve for $x^*(t)$ and $p(t)$ the differential equations

$$\dot{x}^*(t) = f(x^*(t), u^*(t))$$

$$\dot{p}(t) = -\nabla_x H(x^*(t), u^*(t), p(t)),$$

with split boundary conditions:

$$x^*(0) : \text{given}, \quad p(T) = \nabla h(x^*(T)).$$

- The control trajectory is implicitly determined from $x^*(t)$ and $p(t)$ via the equation

$$u^*(t) = \arg \min_{u \in U} H(x^*(t), u, p(t)).$$

EXAMPLE I

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

subject to

$$\dot{x}(t) = u(t), \quad x(0) = \alpha.$$

- Hamiltonian is

$$H(x, u, p) = \sqrt{1 + u^2} + pu,$$

and adjoint equation is $\dot{p}(t) = 0$ with $p(T) = 0$.

- Hence, $p(t) = 0$ for all $t \in [0, T]$, so minimization of the Hamiltonian gives

$$u^*(t) = \arg \min_{u \in \mathcal{R}} \sqrt{1 + u^2} = 0, \quad \text{for all } t \in [0, T].$$

Therefore, $\dot{x}^*(t) = 0$ for all t , implying that $x^*(t)$ is constant. Using the initial condition $x^*(0) = \alpha$, it follows that $x^*(t) = \alpha$ for all t .

EXAMPLE II

- Optimal production problem

$$\text{maximize } \int_0^T (1 - u(t))x(t)dt$$

subject to $0 \leq u(t) \leq 1$ for all t , and

$$\dot{x}(t) = \gamma u(t)x(t), \quad x(0) > 0 : \text{ given.}$$

- Hamiltonian: $H(x, u, p) = (1 - u)x + p\gamma ux$.
- Adjoint equation is

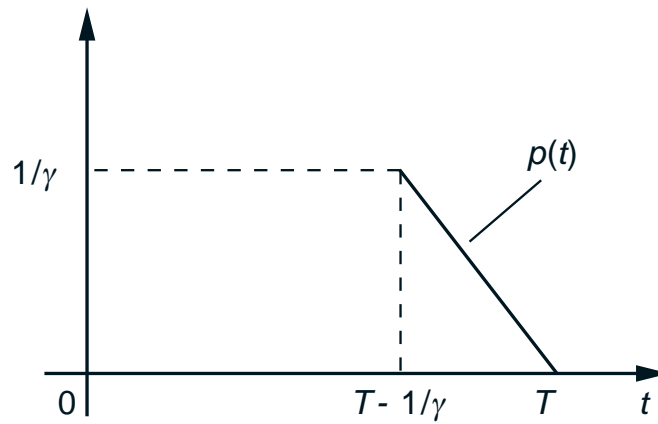
$$\dot{p}(t) = -\gamma u^*(t)p(t) - 1 + u^*(t), \quad p(T) = 0.$$

- Maximization of the Hamiltonian over $u \in [0, 1]$:

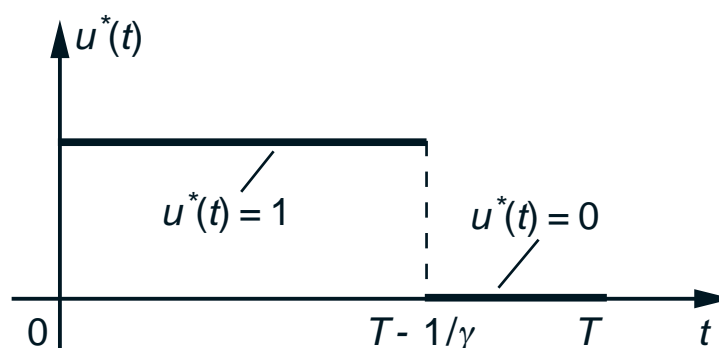
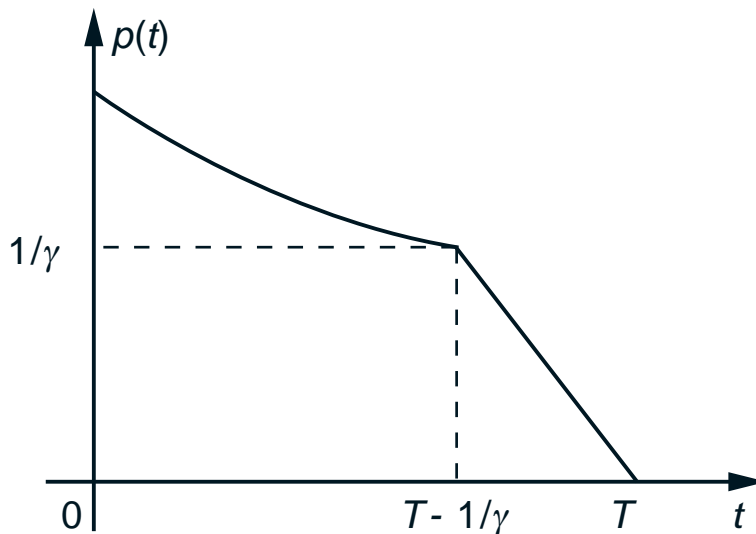
$$u^*(t) = \begin{cases} 0 & \text{if } p(t) < \frac{1}{\gamma}, \\ 1 & \text{if } p(t) \geq \frac{1}{\gamma}. \end{cases}$$

Since $p(T) = 0$, for t close to T , $p(t) < 1/\gamma$ and $u^*(t) = 0$. Therefore, for t near T the adjoint equation has the form $\dot{p}(t) = -1$.

EXAMPLE II (CONTINUED)



- For $t = T - 1/\gamma$, $p(t)$ is equal to $1/\gamma$, so $u^*(t)$ changes to $u^*(t) = 1$.
- Geometrical construction



6.231 DYNAMIC PROGRAMMING

LECTURE 9

LECTURE OUTLINE

- Deterministic continuous-time optimal control
- Variants of the Pontryagin Minimum Principle
- Fixed terminal state
- Free terminal time
- Examples
- Discrete-Time Minimum Principle

REVIEW

- Continuous-time dynamic system

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

- Cost function

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

- $J^*(t, x)$: optimal cost-to-go from x at time t
- HJB equation/Verification theorem: 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 $J^*(T, x) = h(x)$.

- Adjoint equation/vector: To compute an optimal state-control trajectory $\{(u^*(t), x^*(t))\}$ it is enough to know

$$p(t) = \nabla_x J^*(t, x^*(t)), \quad t \in [0, T].$$

- Pontryagin theorem gives an equation for $p(t)$.

NEC. CONDITION: PONTRYAGIN MIN. PRINCIPLE

- Define the Hamiltonian function

$$H(x, u, p) = g(x, u) + p' f(x, u).$$

- **Minimum Principle:** Let $\{u^*(t) \mid t \in [0, T]\}$ be an optimal control trajectory and let $\{x^*(t) \mid t \in [0, T]\}$ be the corresponding state trajectory. Let also $p(t)$ be the solution of the adjoint equation

$$\dot{p}(t) = -\nabla_x H(x^*(t), u^*(t), p(t)),$$

with the boundary condition

$$p(T) = \nabla h(x^*(T)).$$

Then, for all $t \in [0, T]$,

$$u^*(t) = \arg \min_{u \in U} H(x^*(t), u, p(t)).$$

Furthermore, there is a constant C such that

$$H(x^*(t), u^*(t), p(t)) = C, \quad \text{for all } t \in [0, T].$$

VARIATIONS: FIXED TERMINAL STATE

- Suppose that in addition to the initial state $x(0)$, the final state $x(T)$ is given.
- Then the informal derivation of the adjoint equation still holds, but the terminal condition $J^*(T, x) \equiv h(x)$ of the HJB equation is not true anymore.
- In effect,

$$J^*(T, x) = \begin{cases} 0 & \text{if } x = x(T) \\ \infty & \text{otherwise.} \end{cases}$$

So $J^*(T, x)$ cannot be differentiated with respect to x , and the terminal boundary condition $p(T) = \nabla h(x^*(T))$ for the adjoint equation does not hold.

- As compensation, we have the extra condition

$$x(T) : \text{given,}$$

thus maintaining the balance between boundary conditions and unknowns.

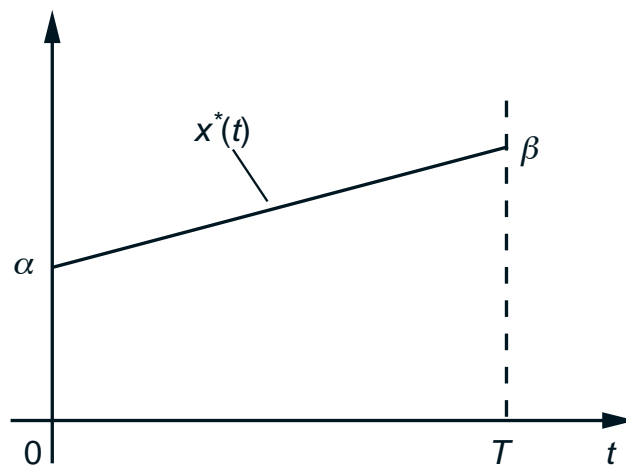
- Generalization: Some components of the terminal state are fixed.

EXAMPLE WITH FIXED TERMINAL STATE

- Consider finding the curve of minimum length connecting two points $(0, \alpha)$ and (T, β) . We have

$$\dot{x}(t) = u(t), \quad x(0) = \alpha, \quad x(T) = \beta,$$

and the cost is $\int_0^T \sqrt{1 + (u(t))^2} dt$.



- The adjoint equation is $\dot{p}(t) = 0$, implying that

$$p(t) = \text{constant}, \quad \text{for all } t \in [0, T].$$

- Minimizing the Hamiltonian $\sqrt{1 + u^2} + p(t)u$:

$$u^*(t) = \text{constant}, \quad \text{for all } t \in [0, T].$$

So optimal $\{x^*(t) \mid t \in [0, T]\}$ is a straight line.

VARIATIONS: FREE TERMINAL TIME

- Initial state and/or the terminal state are given, but the terminal time T is subject to optimization.
- Let $\{(x^*(t), u^*(t)) \mid t \in [0, T]\}$ be an optimal state-control trajectory pair and let T^* be the optimal terminal time. Then $x^*(t), u^*(t)$ would still be optimal if T were fixed at T^* , so

$$u^*(t) = \arg \min_{u \in U} H(x^*(t), u, p(t)), \quad \text{for all } t \in [0, T^*]$$

where $p(t)$ is given by the adjoint equation.

- In addition: $H(x^*(t), u^*(t), p(t)) \equiv 0$ [i.e., for all t , instead of $H(x^*(t), u^*(t), p(t)) \equiv \text{constant}$].
- Justification: We have

$$\nabla_t J^*(t, x^*(t)) \Big|_{t=0} = 0$$

Along the optimal, the HJB equation is

$$\nabla_t J^*(t, x^*(t)) = -H(x^*(t), u^*(t), p(t)), \quad \text{for all } t$$

so $H(x^*(0), u^*(0), p(0)) = 0$.

MINIMUM-TIME EXAMPLE I

- Unit mass moves horizontally: $\ddot{y}(t) = u(t)$, where $y(t)$: position, $u(t)$: force, $u(t) \in [-1, 1]$.
- Given the initial position-velocity $(y(0), \dot{y}(0))$, bring the object to $(y(T), \dot{y}(T)) = (0, 0)$ so that the time of transfer is minimum. Thus, we want to

$$\text{minimize } T = \int_0^T 1 dt.$$

- Let the state variables be

$$x_1(t) = y(t), \quad x_2(t) = \dot{y}(t),$$

so the system equation is

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

- Initial state $(x_1(0), x_2(0))$: given and

$$x_1(T) = 0, \quad x_2(T) = 0.$$

MINIMUM-TIME EXAMPLE II

- If $\{u^*(t) \mid t \in [0, T]\}$ is optimal, $u^*(t)$ must minimize the Hamiltonian for each t , i.e.,

$$u^*(t) = \arg \min_{-1 \leq u \leq 1} [1 + p_1(t)x_2^*(t) + p_2(t)u].$$

Therefore

$$u^*(t) = \begin{cases} 1 & \text{if } p_2(t) < 0, \\ -1 & \text{if } p_2(t) \geq 0. \end{cases}$$

- The adjoint equation is

$$\dot{p}_1(t) = 0, \quad \dot{p}_2(t) = -p_1(t),$$

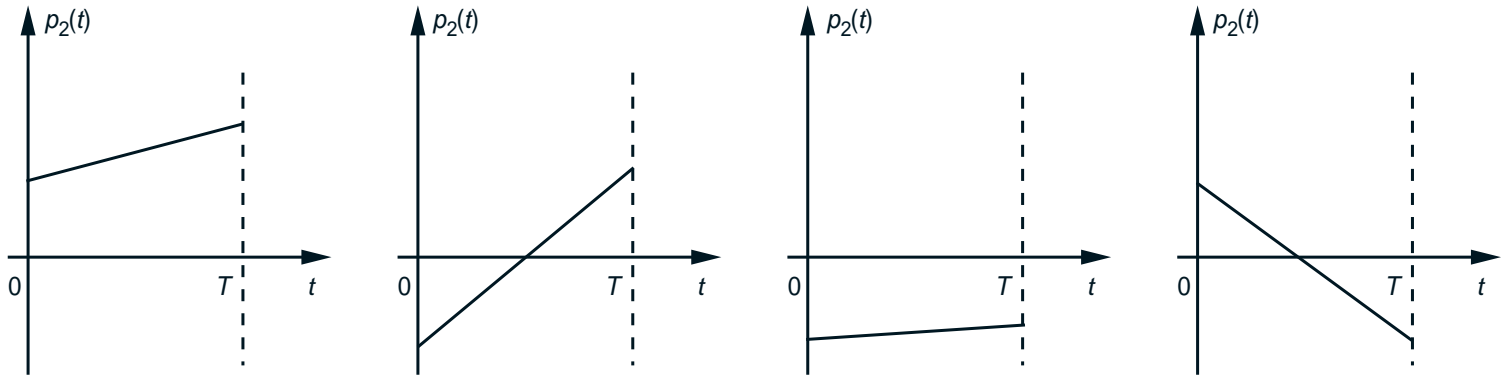
so

$$p_1(t) = c_1, \quad p_2(t) = c_2 - c_1 t,$$

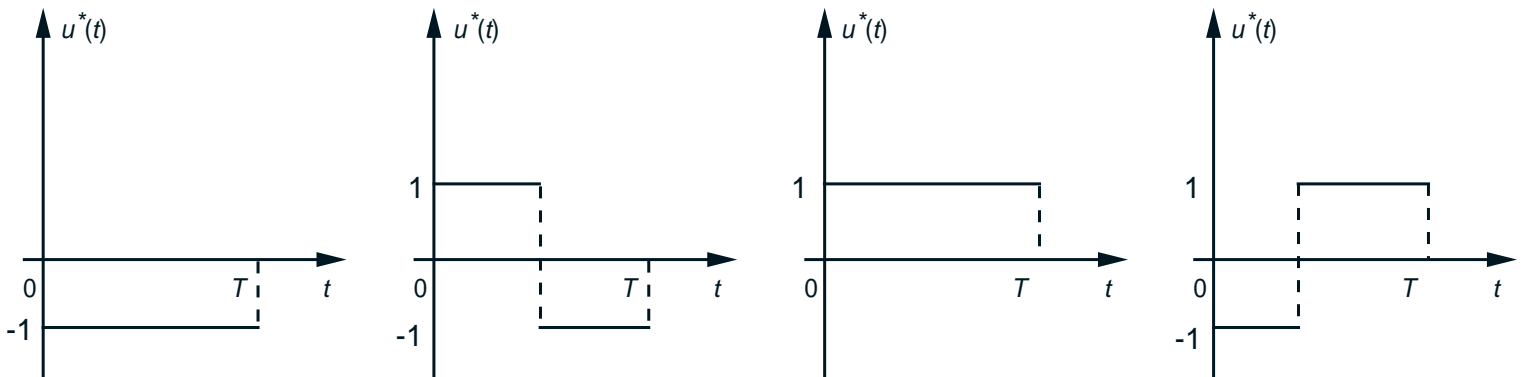
where c_1 and c_2 are constants.

- So $\{p_2(t) \mid t \in [0, T]\}$ switches at most once in going from negative to positive or reversely.

MINIMUM-TIME EXAMPLE III



(a)



(b)

- For $u(t) \equiv \zeta$, where $\zeta = \pm 1$, the system evolves according to

$$x_1(t) = x_1(0) + x_2(0)t + \frac{\zeta}{2}t^2, \quad x_2(t) = x_2(0) + \zeta t.$$

Eliminating the time t , we see that for all t

$$x_1(t) - \frac{1}{2\zeta} (x_2(t))^2 = x_1(0) - \frac{1}{2\zeta} (x_2(0))^2.$$

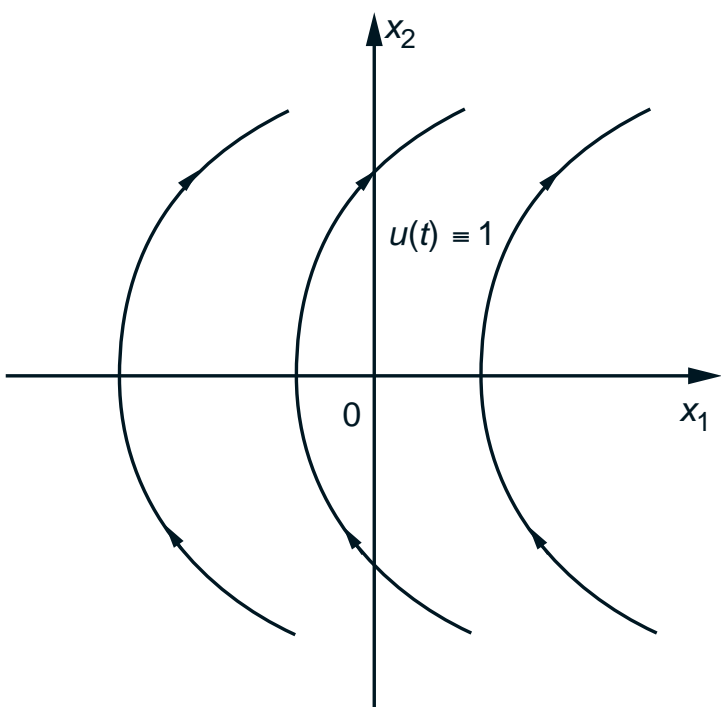
MINIMUM-TIME EXAMPLE IV

- For intervals where $u(t) \equiv 1$, the system moves along the curves

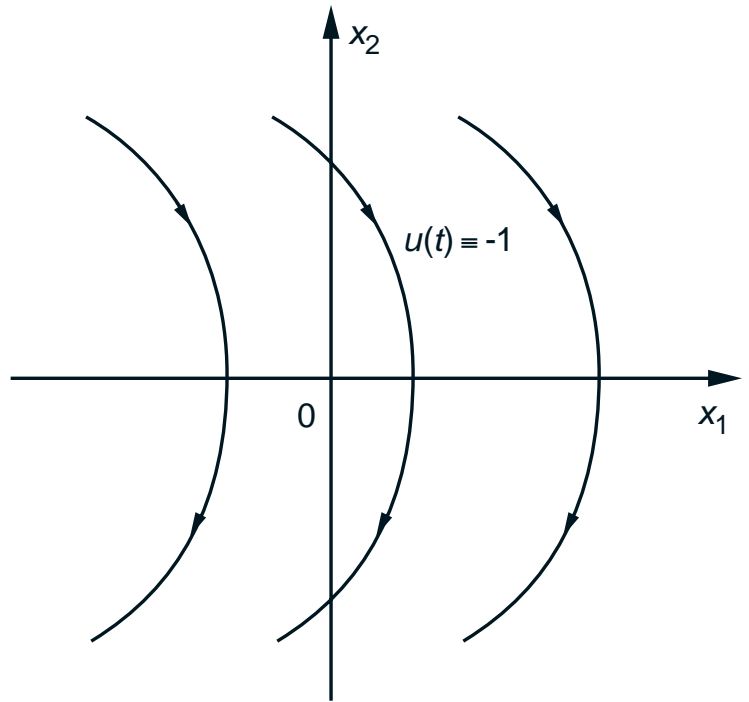
$$x_1(t) - \frac{1}{2}(x_2(t))^2 : \text{constant.}$$

- For intervals where $u(t) \equiv -1$, the system moves along the curves

$$x_1(t) + \frac{1}{2}(x_2(t))^2 : \text{constant.}$$



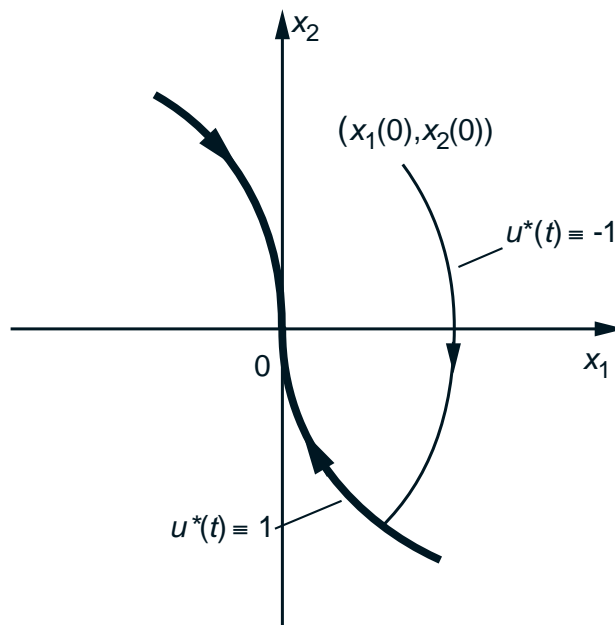
(a)



(b)

MINIMUM-TIME EXAMPLE V

- To bring the system from the initial state $x(0)$ to the origin with at most one switch, we use the following switching curve.



- If the initial state lies *above* the switching curve, use $u^*(t) \equiv -1$ until the state hits the switching curve; then use $u^*(t) \equiv 1$.
- If the initial state lies *below* the switching curve, use $u^*(t) \equiv 1$ until the state hits the switching curve; then use $u^*(t) \equiv -1$.
- If the initial state lies on the top (bottom) part of the switching curve, use $u^*(t) \equiv -1$ [$u^*(t) \equiv 1$, respectively].

DISCRETE-TIME MINIMUM PRINCIPLE

- Minimize $J(u) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$, subject to $u_k \in U_k \subset \mathfrak{R}^m$, with U_k : convex, and

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, \dots, N-1, \quad x_0 : \text{ given.}$$

- Introduce Hamiltonian function

$$H_k(x_k, u_k, p_{k+1}) = g_k(x_k, u_k) + p'_{k+1} f_k(x_k, u_k)$$

- Suppose $\{(u_k^*, x_{k+1}^*) \mid k = 0, \dots, N-1\}$ are optimal. Then for all k ,

$$\nabla_{u_k} H_k(x_k^*, u_k^*, p_{k+1})' (u_k - u_k^*) \geq 0, \quad \text{for all } u_k \in U_k,$$

where p_1, \dots, p_N are obtained from

$$p_k = \nabla_{x_k} f_k \cdot p_{k+1} + \nabla_{x_k} g_k,$$

with the terminal condition $p_N = \nabla g_N(x_N^*)$.

- If, in addition, the Hamiltonian H_k is a convex function of u_k for any fixed x_k and p_{k+1} , we have

$$u_k^* = \arg \min_{u_k \in U_k} H_k(x_k^*, u_k, p_{k+1}), \quad \text{for all } k.$$

DERIVATION

- We develop an expression for the gradient $\nabla J(u)$. We have, using the chain rule,

$$\begin{aligned}\nabla_{u_k} J(u) &= \nabla_{u_k} f_k \cdot \nabla_{x_{k+1}} f_{k+1} \cdots \nabla_{x_{N-1}} f_{N-1} \cdot \nabla g_N \\ &\quad + \nabla_{u_k} f_k \cdot \nabla_{x_{k+1}} f_{k+1} \cdots \nabla_{x_{N-2}} f_{N-2} \cdot \nabla_{x_{N-1}} g_{N-1} \\ &\quad \dots \\ &\quad + \nabla_{u_k} f_k \cdot \nabla_{x_{k+1}} g_{k+1} \\ &\quad + \nabla_{u_k} g_k,\end{aligned}$$

where all gradients are evaluated along u and the corresponding state trajectory.

- Introduce the discrete-time adjoint equation

$$p_k = \nabla_{x_k} f_k \cdot p_{k+1} + \nabla_{x_k} g_k, \quad k = 1, \dots, N-1,$$

with terminal condition $p_N = \nabla g_N$.

- Verify that, for all k ,

$$\nabla_{u_k} J(u_0, \dots, u_{N-1}) = \nabla_{u_k} H_k(x_k, u_k, p_{k+1})$$

6.231 DYNAMIC PROGRAMMING

LECTURE 10

LECTURE OUTLINE

- Problems with imperfect state info
- Reduction to the perfect state info case
- Machine repair example

BASIC PROBLEM WITH 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

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

- 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

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

subject to the equations

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

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

EXAMPLE: MULTIAccess COMMUNICATION I

- Collection of transmitting stations sharing a common channel, are synchronized to transmit packets of data at integer times.
- x_k : backlog at the beginning of slot k .
- a_k : random number of packet arrivals in slot k .
- t_k : the number of packets transmitted in slot k .

$$x_{k+1} = x_k + a_k - t_k,$$

- At k th slot, each of the x_k packets in the system is transmitted with probability u_k (common for all packets). If two or more packets are transmitted simultaneously, they collide.
- So $t_k = 1$ (a success) with probability $x_k u_k (1 - u_k)^{x_k - 1}$, and $t_k = 0$ (idle or collision) otherwise.
- Imperfect state info: The stations can observe the channel and determine whether in any one slot there was a collision (two or more packets), a success (one packet), or an idle (no packets).

EXAMPLE: MULTIACCESS COMMUNICATION II

- Information vector at time k : The entire history (up to k) of successes, idles, and collisions. Mathematically, z_{k+1} , the observation at the end of the k th slot, is

$$z_{k+1} = v_{k+1}$$

where v_{k+1} yields an idle with probability $(1 - u_k)^{x_k}$, a success with probability $x_k u_k (1 - u_k)^{x_k - 1}$, and a collision otherwise.

- If we had perfect state information, the DP algorithm would be

$$J_k(x_k) = g_k(x_k) + \min_{0 \leq u_k \leq 1} E \left\{ p(x_k, u_k) J_{k+1}(x_k + a_k - 1) + (1 - p(x_k, u_k)) J_{k+1}(x_k + a_k) \right\},$$

$p(x_k, u_k)$ is the success probability $x_k u_k (1 - u_k)^{x_k - 1}$.

- The optimal (perfect state information) policy would be to select the value of u_k that maximizes $p(x_k, u_k)$, so $\mu_k(x_k) = \frac{1}{x_k}$, for all $x_k \geq 1$.
- Imperfect state info problem is much harder.

REFORMULATION AS A PERFECT INFO PROBLEM

- 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

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

so the cost per stage of the new system is

$$\tilde{g}_k(I_k, u_k) = \underset{x_k, w_k}{E} \left\{ g_k(x_k, u_k, w_k) \mid I_k, u_k \right\}$$

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 $k = N - 1$,

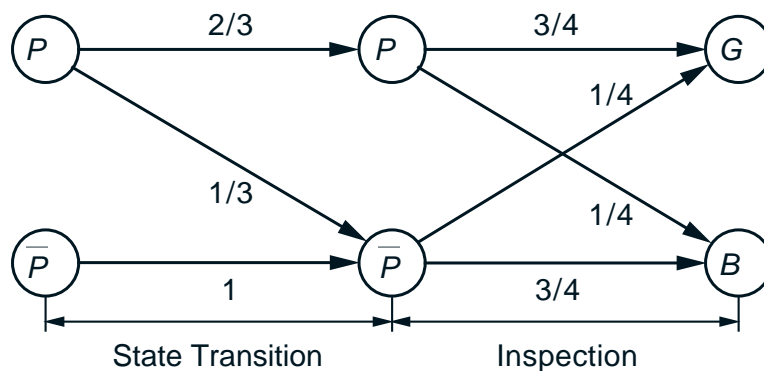
$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \left[E_{x_{N-1}, w_{N-1}} \left\{ g_N(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})) + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) \mid I_{N-1}, u_{N-1} \right\} \right],$$

- The optimal cost J^* is given by

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

MACHINE REPAIR EXAMPLE I

- A machine can be in one of two states denoted P (good state) and \bar{P} (bad state).
- At the end of each period the machine is inspected.
- Two possible inspection outcomes: G (probably good state) and B (probably bad state).
- Transition probabilities:



- Possible actions after each inspection:
 - C : Continue operation of the machine.
 - S : Stop the machine, determine its state, and if in \bar{P} bring it back to the good state P .
- Cost per stage:
 - $g(P, C) = 0, g(P, S) = 1, g(\bar{P}, C) = 2, g(\bar{P}, S) = 1.$

MACHINE REPAIR EXAMPLE II

- The information vector at times 0 and 1 is

$$I_0 = z_0, \quad I_1 = (z_0, z_1, u_0),$$

and we seek functions $\mu_0(I_0), \mu_1(I_1)$ that minimize

$$E_{\substack{x_0, w_0, w_1 \\ v_0, v_1}} \left\{ g(x_0, \mu_0(z_0)) + g(x_1, \mu_1(z_0, z_1, \mu_0(z_0))) \right\}.$$

- DP algorithm: Start with $J_2(I_2) = 0$. For $k = 0, 1$, take the min over the two actions, C and S,

$$\begin{aligned}
 J_k(I_k) = \min & \left[P(x_k = P \mid I_k, C)g(P, C) \right. \\
 & + P(x_k = \bar{P} \mid I_k, C)g(\bar{P}, C) \\
 & + E_{z_{k+1}} \left\{ J_{k+1}(I_k, C, z_{k+1}) \mid I_k, C \right\}, \\
 & P(x_k = P \mid I_k, S)g(P, S) \\
 & + P(x_k = \bar{P} \mid I_k, S)g(\bar{P}, S) \\
 & \left. + E_{z_{k+1}} \left\{ J_{k+1}(I_k, S, z_{k+1}) \mid I_k, S \right\} \right]
 \end{aligned}$$

MACHINE REPAIR EXAMPLE III

• *Last Stage*: Compute $J_1(I_1)$ for each of the eight possible information vectors $I_1 = (z_0, z_1, u_0)$. We have

cost of $C = 2 \cdot P(x_1 = \bar{P} \mid I_1)$, cost of $S = 1$,

and therefore $J_1(I_1) = \min[2P(x_1 = \bar{P} \mid I_1), 1]$.
The probabilities $P(x_1 = \bar{P} \mid I_1)$ are computed using Bayes' rule:

(1) For $I_1 = (G, G, S)$

$$\begin{aligned} P(x_1 = \bar{P} \mid G, G, S) &= \frac{P(x_1 = \bar{P}, G, G, S)}{P(G, G, S)} \\ &= \frac{\frac{1}{3} \cdot \frac{1}{4} \cdot \left(\frac{2}{3} \cdot \frac{3}{4} + \frac{1}{3} \cdot \frac{1}{4}\right)}{\left(\frac{2}{3} \cdot \frac{3}{4} + \frac{1}{3} \cdot \frac{1}{4}\right)^2} = \frac{1}{7}. \end{aligned}$$

Hence

$$J_1(G, G, S) = \frac{2}{7}, \quad \mu_1^*(G, G, S) = C.$$

MACHINE REPAIR EXAMPLE IV

(2) For $I_1 = (B, G, S)$

$$P(x_1 = \bar{P} \mid B, G, S) = P(x_1 = \bar{P} \mid G, G, S) = \frac{1}{7},$$

$$J_1(B, G, S) = \frac{2}{7}, \quad \mu_1^*(B, G, S) = C.$$

(3) For $I_1 = (G, B, S)$

$$\begin{aligned} P(x_1 = \bar{P} \mid G, B, S) &= \frac{P(x_1 = \bar{P}, G, B, S)}{P(G, B, S)} \\ &= \frac{\frac{1}{3} \cdot \frac{3}{4} \cdot \left(\frac{2}{3} \cdot \frac{3}{4} + \frac{1}{3} \cdot \frac{1}{4} \right)}{\left(\frac{2}{3} \cdot \frac{1}{4} + \frac{1}{3} \cdot \frac{3}{4} \right) \left(\frac{2}{3} \cdot \frac{3}{4} + \frac{1}{3} \cdot \frac{1}{4} \right)} \\ &= \frac{3}{5}, \end{aligned}$$

$$J_1(G, B, S) = 1, \quad \mu_1^*(G, B, S) = S.$$

- Similarly, for all possible I_1 , we compute $J_1(I_1)$, and $\mu_1^*(I_1)$, which is to continue ($u_1 = C$) if the last inspection was G , and to stop otherwise.

MACHINE REPAIR EXAMPLE V

- *First Stage*: Compute $J_0(I_0)$ for each of the two possible information vectors $I_0 = (G)$, $I_0 = (B)$. We have

$$\begin{aligned}\text{cost of } C &= 2P(x_0 = \bar{P} \mid I_0, C) + E_{z_1} \{ J_1(I_0, z_1, C) \mid I_0, C \} \\ &= 2P(x_0 = \bar{P} \mid I_0, C) + P(z_1 = G \mid I_0, C)J_1(I_0, G, C) \\ &\quad + P(z_1 = B \mid I_0, C)J_1(I_0, B, C),\end{aligned}$$

$$\begin{aligned}\text{cost of } S &= 1 + E_{z_1} \{ J_1(I_0, z_1, S) \mid I_0, S \} \\ &= 1 + P(z_1 = G \mid I_0, S)J_1(I_0, G, S) \\ &\quad + P(z_1 = B \mid I_0, S)J_1(I_0, B, S),\end{aligned}$$

using the values of J_1 from the previous stage.

- We have

$$J_0(I_0) = \min[\text{cost of } C, \text{cost of } S]$$

- The optimal cost is

$$J^* = P(G)J_0(G) + P(B)J_0(B).$$

6.231 DYNAMIC PROGRAMMING

LECTURE 11

LECTURE OUTLINE

- Review of DP for imperfect state info
- Linear quadratic problems
- Separation of estimation and control

REVIEW: PROBLEM WITH IMPERFECT STATE INFO

- 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

- Reformulate to perfect state info problem, and write the DP algorithm:

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

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

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

- The optimal cost J^* is given by

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

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:

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

- 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 $N - 1$):

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

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

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

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

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

where

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

DP ALGORITHM II

- Substituting in the DP algorithm

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

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

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

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

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

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

DP ALGORITHM III

- DP equation for period $N - 2$:

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

- 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

- For every k , there is a function M_k such that we have

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

independently of the policy being used.

- Using the lemma,

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

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

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

and is independent of u_{N-2} .

- So minimization in the DP algorithm yields

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

FINAL RESULT

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

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

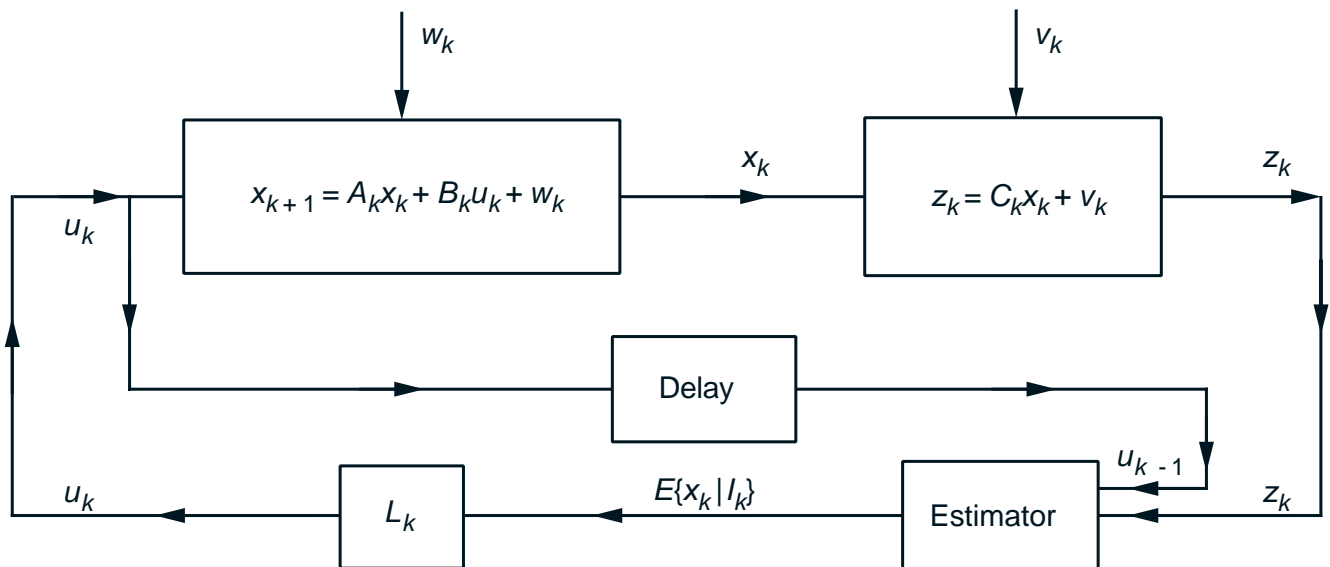
where L_k is the same as for perfect state info:

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

with K_k generated from $K_N = Q_N$, using

$$K_k = A_k' K_{k+1} A_k - P_k + Q_k,$$

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



SEPARATION INTERPRETATION

- The optimal controller can be decomposed into
 - (a) An *estimator*, which uses the data to generate the conditional expectation $\mathbf{E}\{x_k \mid I_k\}$.
 - (b) An *actuator*, which multiplies $\mathbf{E}\{x_k \mid I_k\}$ by the gain matrix L_k and applies the control input $u_k = L_k \mathbf{E}\{x_k \mid I_k\}$.
- Generically the estimate \hat{x} of a random vector x given some information (random vector) I , which minimizes the mean squared error $\mathbf{E}_x\{\|x - \hat{x}\|^2 \mid I\}$ is $\mathbf{E}\{x \mid I\}$ (expand the quadratic form and set to zero the derivative with respect to \hat{x}).
- The estimator portion of the optimal controller is an optimal solution of the problem of estimating the state x_k assuming the control is not subject to choice.
- The actuator portion is an optimal solution of the control problem assuming perfect state information prevails.

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, $\mathbf{E}\{x_k \mid I_k\}$ 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 $\mathbf{E}\{x_k \mid I_k\}$ maybe very difficult, so a suboptimal solution is typically used.
- Most common suboptimal controller: Replace $\mathbf{E}\{x_k \mid I_k\}$ 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 12

LECTURE OUTLINE

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

REVIEW: PROBLEM WITH IMPERFECT STATE INFO

- 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 $P_{X_K|I_K}$

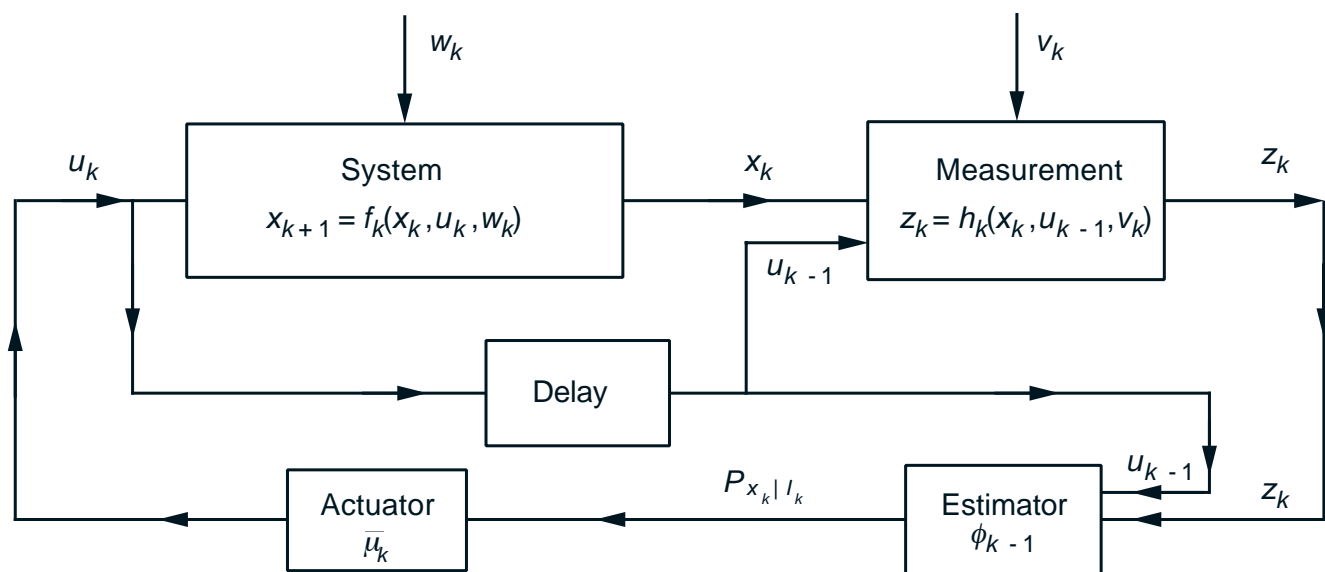
- It turns out that $P_{x_k|I_k}$ is generated recursively by a dynamic system (estimator) of the form

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

for a suitable function Φ_k

- DP algorithm can be written as

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



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

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

SEARCH PROBLEM (CONTINUED)

- DP algorithm

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

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

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

$$\bar{J}_k(p_k) = 0, \quad \text{for all } p_k \leq \frac{C}{\beta V}$$

- 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)

FINITE-STATE SYSTEMS

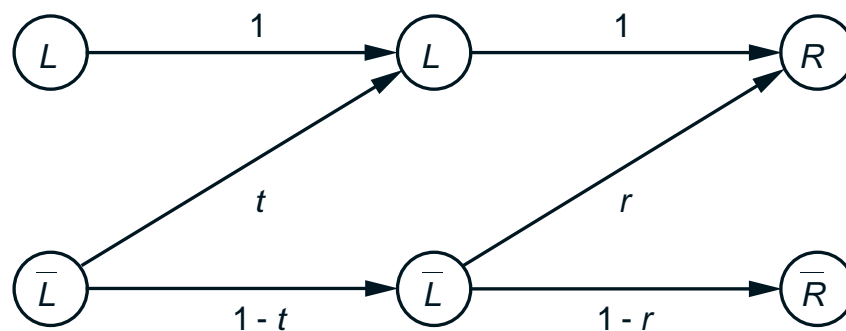
- Suppose the system is a finite-state Markov chain, with states $1, \dots, n$.
- Then the conditional probability distribution $P_{x_k | I_k}$ is a vector

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

- 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, 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 and also (moderately) useful in practice.

INSTRUCTION EXAMPLE

- 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.
- The test has two possible outcomes: R : Student gives a correct answer, or \bar{R} : Student gives an incorrect answer.
- Probabilistic structure



- Cost of instruction is 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

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

- Using Bayes' rule we can obtain

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

- 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

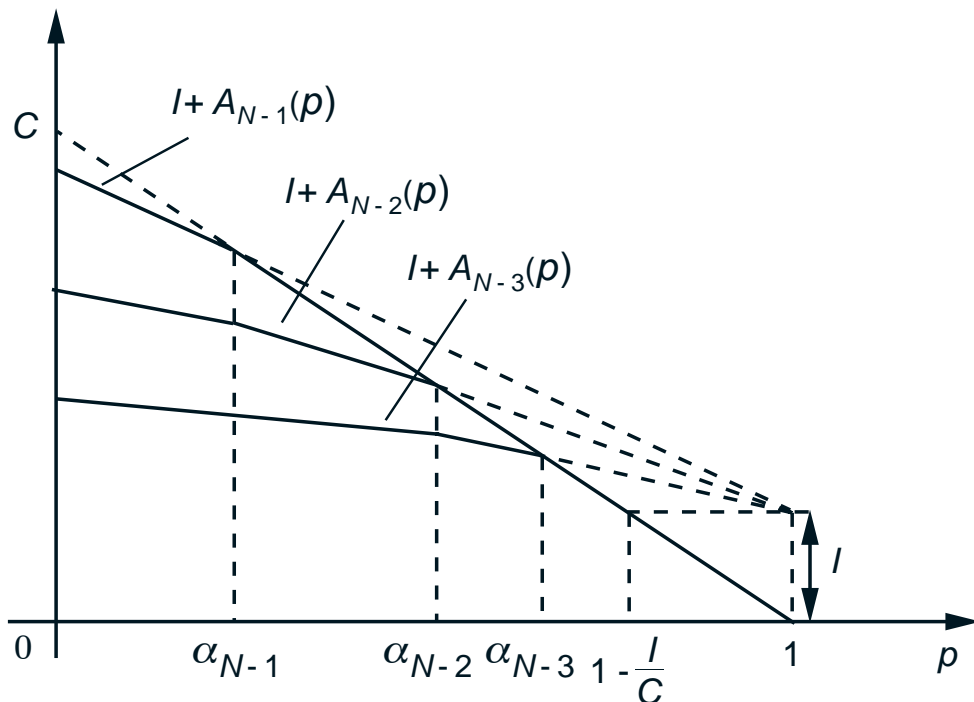
$$\bar{J}_k(p_k) = \min[(1 - p_k)C, I + A_k(p_k)],$$

where

$$A_k(p_k) = P(z_{k+1} = R | I_k) \bar{J}_{k+1}(\Phi(p_k, R)) \\ + P(z_{k+1} = \bar{R} | I_k) \bar{J}_{k+1}(\Phi(p_k, \bar{R}))$$

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

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



6.231 DYNAMIC PROGRAMMING

LECTURE 13

LECTURE OUTLINE

- Suboptimal control
- Certainty equivalent control
- Implementations and approximations
- Issues in adaptive control

PRACTICAL DIFFICULTIES OF DP

- The curse of modeling
- 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
- 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

CERTAINTY EQUIVALENT CONTROL (CEC)

- Replace the stochastic problem with a deterministic problem
- At each time k , the uncertain quantities are fixed at some “typical” values
- Implementation for an imperfect info problem.
At each time k :

- (1) Compute a state estimate $\bar{x}_k(I_k)$ given the current information vector I_k .
- (2) Fix the $w_i, i \geq k$, at some $\bar{w}_i(x_i, u_i)$. Solve the deterministic problem:

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

subject to $x_k = \bar{x}_k(I_k)$ and for $i \geq k$,

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

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

ALTERNATIVE IMPLEMENTATION

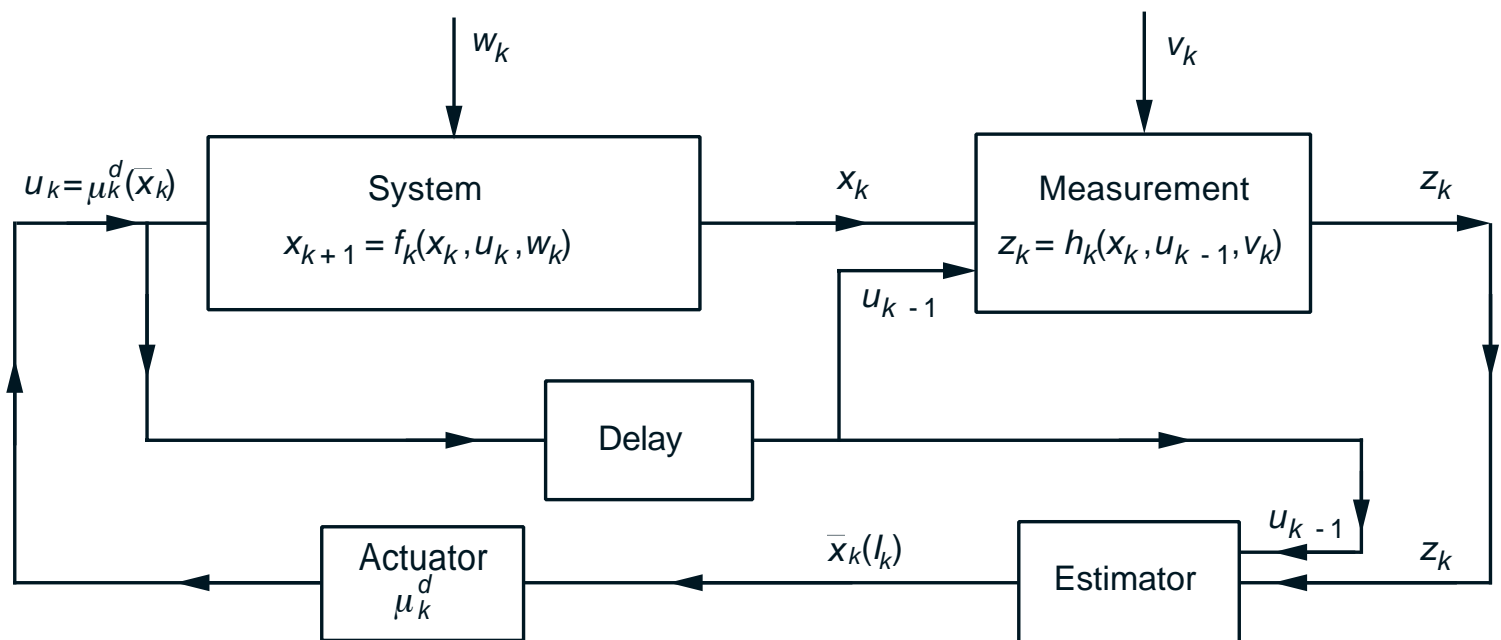
- Let $\{\mu_0^d(x_0), \dots, \mu_{N-1}^d(x_{N-1})\}$ be an optimal controller obtained from the DP algorithm for the deterministic problem

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

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

The CEC applies at time k the control input

$$\tilde{\mu}_k(I_k) = \mu_k^d(\bar{x}_k(I_k))$$



CEC WITH HEURISTICS

- Solve the “deterministic equivalent” problem using a heuristic/suboptimal policy
- Improved version of this idea: At time k minimize the stage k cost and plus the heuristic cost of the remaining stages, i.e., apply at time k a control \tilde{u}_k that minimizes over $u_k \in U_k(x_k)$

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

where H_{k+1} is the cost-to-go function corresponding to the heuristic.

- This an example of an important suboptimal control idea:

Minimize at each stage k the sum of approximations to the current stage cost and the optimal cost-to-go.

- This is a central idea in several other suboptimal control schemes, such as limited lookahead, and rollout algorithms.

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 (discussed in Ch. 5) 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).

SYSTEMS WITH UNKNOWN PARAMETERS

- Let the system be of the form

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

where θ is a vector of unknown parameters with a given a priori probability distribution.

- To formulate this into the standard framework, introduce a state variable $y_k = \theta$ and the system

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

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

- Since $y_k = \theta$ is unobservable, we have a problem of imperfect state information even if the controller knows the state x_k exactly.
- Consider a partially stochastic CEC. If for a fixed parameter vector θ , we can compute the corresponding optimal policy $\{\mu_0^*(I_0, \theta), \dots, \mu_{N-1}^*(I_{N-1}, \theta)\}$ a partially stochastic CEC applies $\mu_k^*(I_k, \hat{\theta}_k)$, where $\hat{\theta}_k$ is some estimate of θ .

THE PROBLEM OF IDENTIFIABILITY

- Suppose we consider two phases:
 - A parameter identification phase (compute an estimate $\hat{\theta}$ of θ)
 - A control phase (apply control that would be optimal if $\hat{\theta}$ were true).
- A fundamental difficulty: the control process may make some of the unknown parameters invisible to the identification process.
- Example: Consider the scalar system

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

with the cost $E \left\{ \sum_{k=1}^N (x_k)^2 \right\}$. If a and b are known, the optimal control law is $\mu_k^*(x_k) = -(a/b)x_k$.

- If a and b are not known and we try to estimate them while applying some nominal control law $\tilde{\mu}_k(x_k) = \gamma x_k$, the closed-loop system is

$$x_{k+1} = (a + b\gamma)x_k + w_k,$$

so identification can at best find $(a + b\gamma)$ but not the values of both a and b .

CEC AND IDENTIFIABILITY I

- Suppose we have $P\{x_{k+1} \mid x_k, u_k, \theta\}$ and we use a stationary control law:

$$\hat{\mu}_k(I_k) = \mu^*(x_k, \hat{\theta}_k), \quad k = 0, 1, \dots$$

There are three systems of interest:

- (a) The system (perhaps falsely) believed by the controller to be true, which evolves probabilistically according to

$$P\{x_{k+1} \mid x_k, \mu^*(x_k, \hat{\theta}_k), \hat{\theta}_k\}.$$

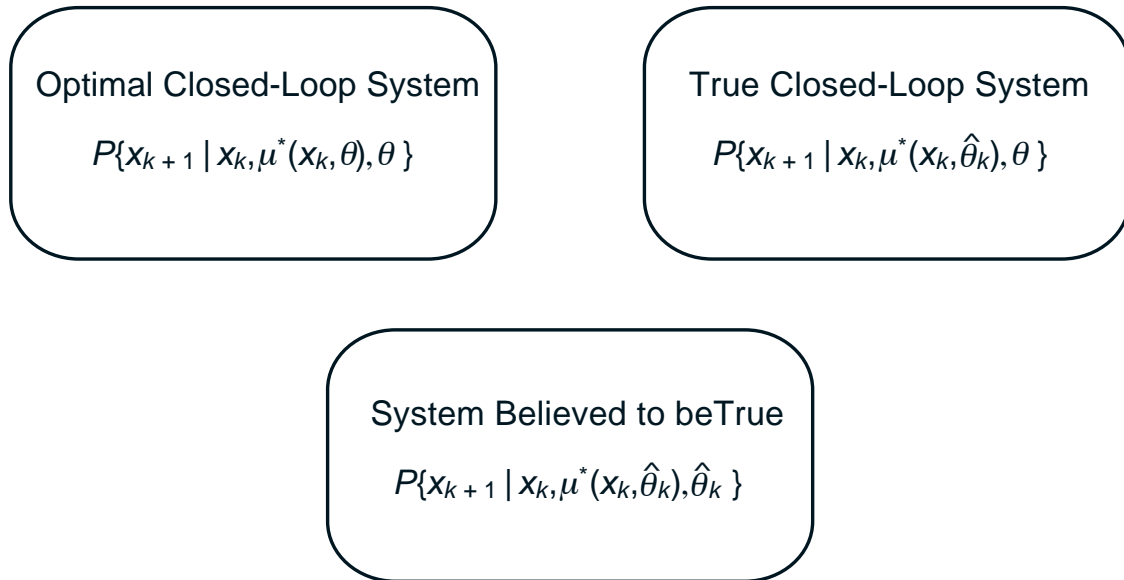
- (b) The true closed-loop system, which evolves probabilistically according to

$$P\{x_{k+1} \mid x_k, \mu^*(x_k, \hat{\theta}_k), \theta\}.$$

- (c) The optimal closed-loop system that corresponds to the true value of the parameter, which evolves probabilistically according to

$$P\{x_{k+1} \mid x_k, \mu^*(x_k, \theta), \theta\}.$$

CEC AND IDENTIFIABILITY II



- There is a built-in mechanism for the parameter estimates to converge to a wrong value
- Assume that for some $\hat{\theta} \neq \theta$ and all $x_{k+1}, x_k,$

$$P\{x_{k+1} | x_k, \mu^*(x_k, \hat{\theta}), \hat{\theta}\} = P\{x_{k+1} | x_k, \mu^*(x_k, \hat{\theta}), \theta\}$$

i.e., there is a false value of parameter for which the system under closed-loop control looks exactly as if the false value were true.

- Then, if the controller estimates at some time the parameter to be $\hat{\theta}$, subsequent data will tend to reinforce this erroneous estimate.

6.231 DYNAMIC PROGRAMMING

LECTURE 14

LECTURE OUTLINE

- Limited lookahead policies
- Performance bounds
- Computational aspects
- Problem approximation approach
- Vehicle routing example
- Heuristic cost-to-go approximation
- Computer chess

LIMITED LOOKAHEAD POLICIES

- *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

- 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 ,

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

Then

$$\bar{J}_k(x_k) \leq \hat{J}_k(x_k) \leq \tilde{J}_k(x_k)$$

- Important application: When \tilde{J}_k is the cost-to-go of some heuristic policy (then the 1SL policy is called the *rollout* policy)
- Bound can be extended to 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 the methodology of stochastic programming.
- The choice of the approximating functions \tilde{J}_k is critical, and is calculated with a variety of methods.
- Some approaches:
 - (a) *Problem Approximation*: Approximate the optimal cost-to-go with some cost derived from a related but simpler problem
 - (b) *Heuristic 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

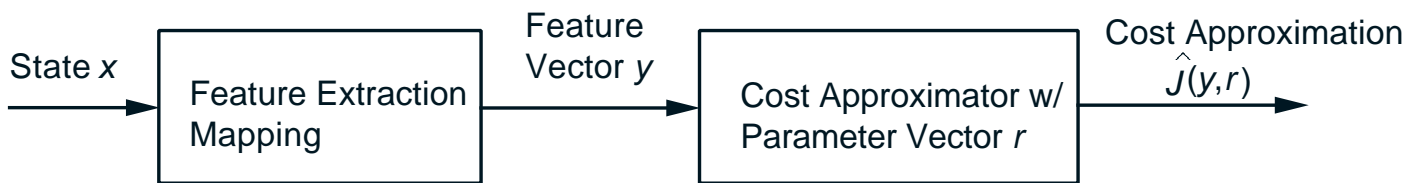
- Many (problem-dependent) possibilities
 - Replace uncertain quantities by nominal values, or simplify the calculation of expected values by limited simulation
 - Simplify difficult constraints or dynamics
- Example of *enforced decomposition*: Route m vehicles that move over a graph. Each node has 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

HEURISTIC 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 simulator is used, particularly when there is no mathematical model of the system.

APPROXIMATION ARCHITECTURES

- Broadly divided in linear and nonlinear [depending on whether $\tilde{J}(x, r)$ depends linearly or nonlinearly on r]
- Linear architectures are easier to train, but nonlinear ones (e.g., neural networks) are richer.
- Architectures based on feature extraction

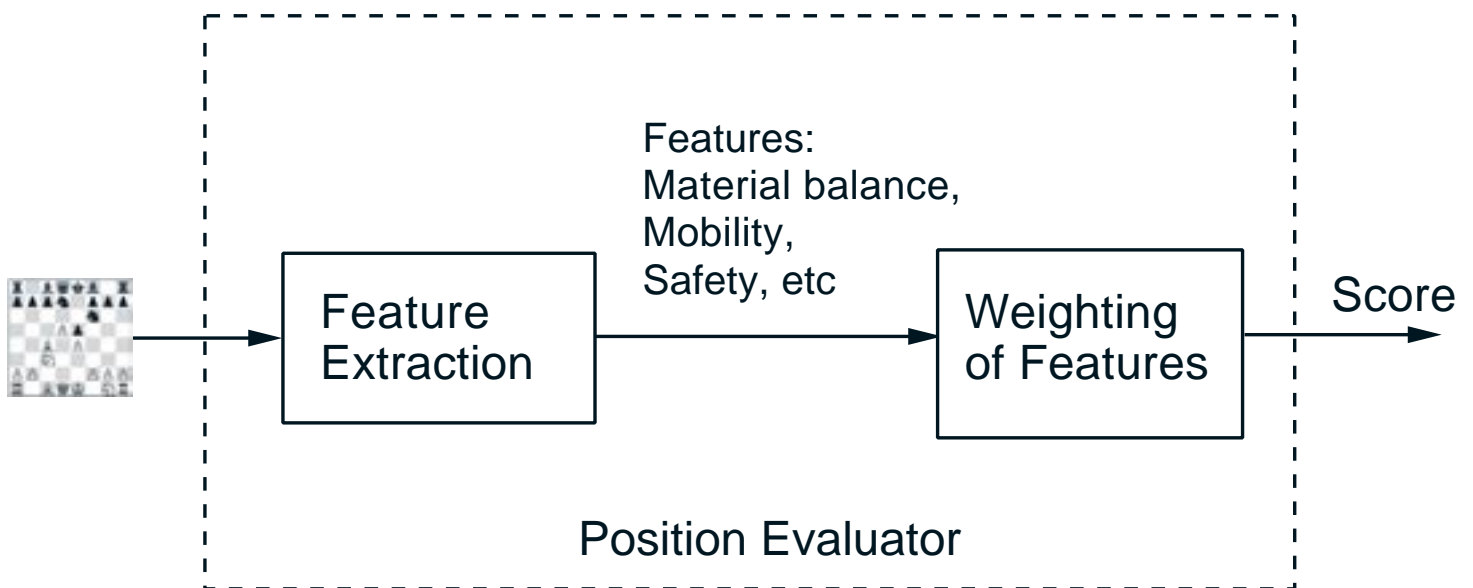


- Ideally, the features will encode much of the nonlinearity that is inherent in the cost-to-go approximated, and the approximation may be quite accurate without a complicated architecture.
- With a well-chosen feature vector $y(x)$, we can use a linear architecture

$$\tilde{J}(x, r) = \hat{J}(y(x), r) = \sum_{i=1}^m r_i y_i(x)$$

COMPUTER CHESS I

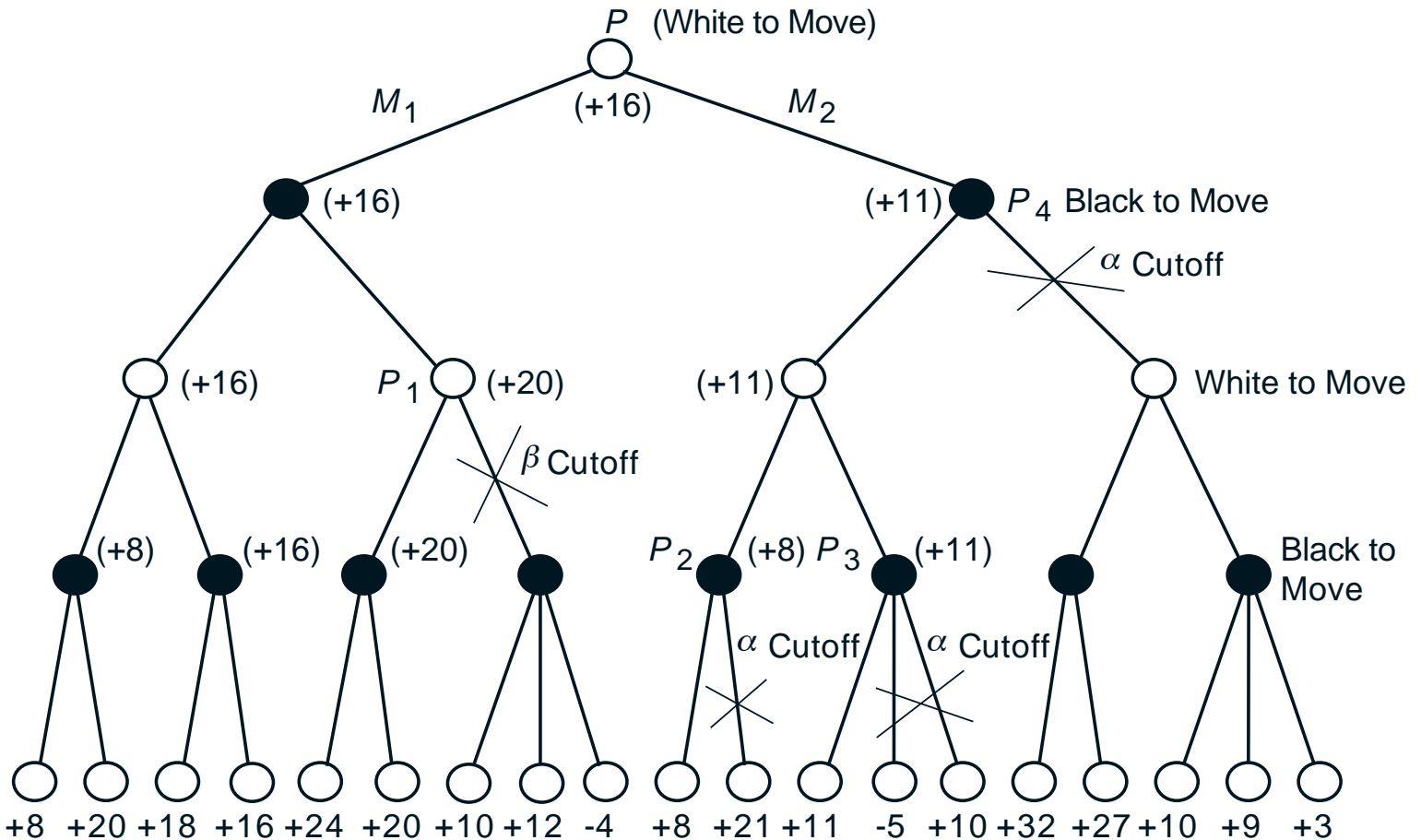
- Programs use a feature-based position evaluator that assigns a score to each move/position



- Most often the weighting of features is linear but multistep lookahead is involved.
- Most often the training is done by trial and error.
- Additional features:
 - Depth first search
 - Variable depth search when dynamic positions are involved
 - Alpha-beta pruning

COMPUTER CHESS II

- Multistep lookahead tree



- Alpha-beta pruning: As the move scores are evaluated by depth-first search, branches whose consideration (based on the calculations so far) cannot possibly change the optimal move are neglected

6.231 DYNAMIC PROGRAMMING

LECTURE 15

LECTURE OUTLINE

- Rollout algorithms
- Cost improvement property
- Discrete deterministic problems
- Sequential consistency and greedy algorithms
- Sequential improvement

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 , with 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*: Questions should be answered in decreasing order of the “index of preference” $p_i v_i / (1 - p_i)$.
- 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.

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

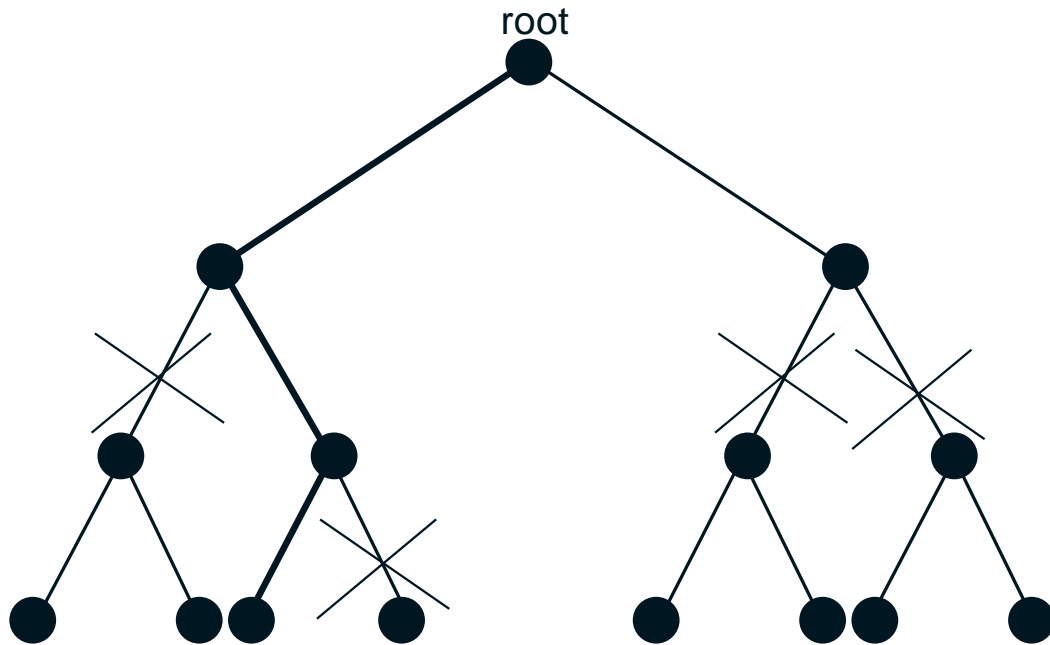
- We claim that $\bar{J}_k(x_k) \leq H_k(x_k)$ for all x_k and k
- Proof by induction: We have $\bar{J}_N(x_N) = H_N(x_N)$ for all x_N . Assume that

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

Then, for all x_k

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

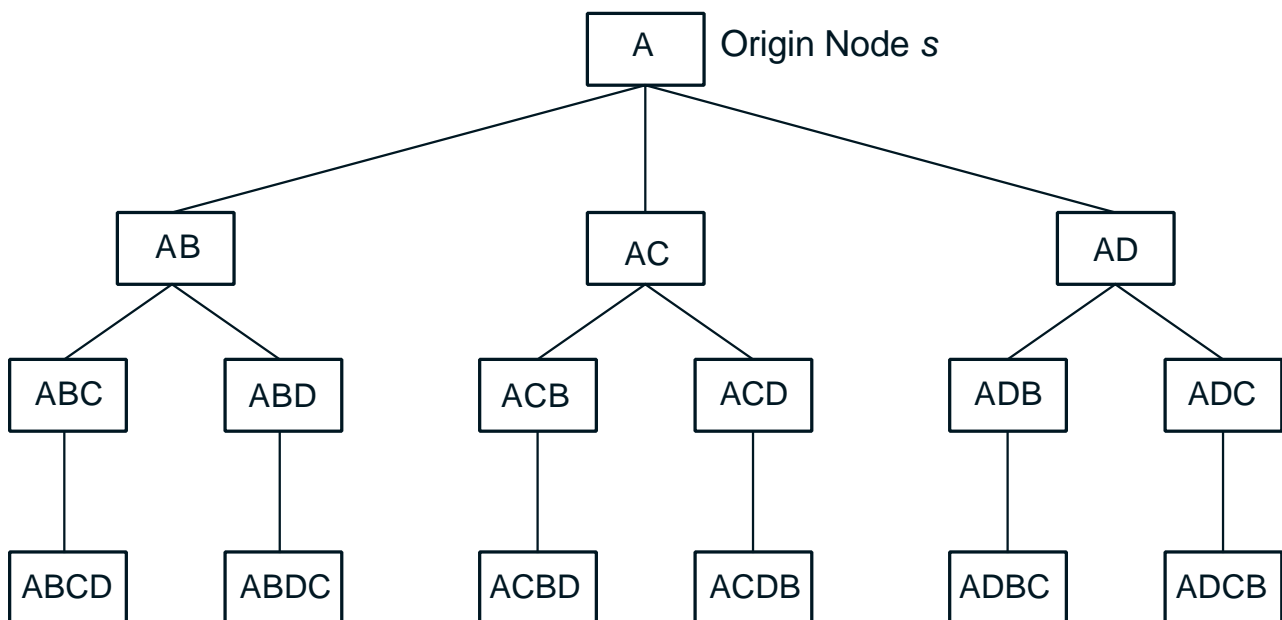
EXAMPLE: THE BREAKTHROUGH PROBLEM



- 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.
- 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.

DISCRETE DETERMINISTIC PROBLEMS

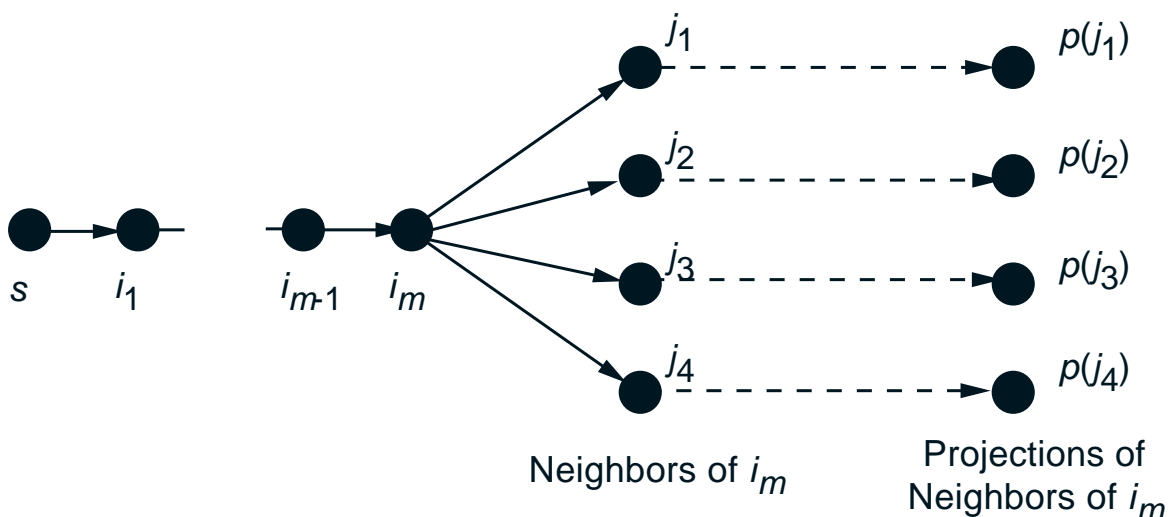
- Any discrete optimization problem (with finite number of choices/feasible solutions) can be represented as a sequential decision process by using a tree.
- The leaves of the tree correspond to the feasible solutions.
- The problem can be solved by DP, starting from the leaves and going back towards the root.
- Example: Traveling salesman problem. Find a minimum cost tour that goes exactly once through each of N cities.



Traveling salesman problem with four cities A, B, C, D

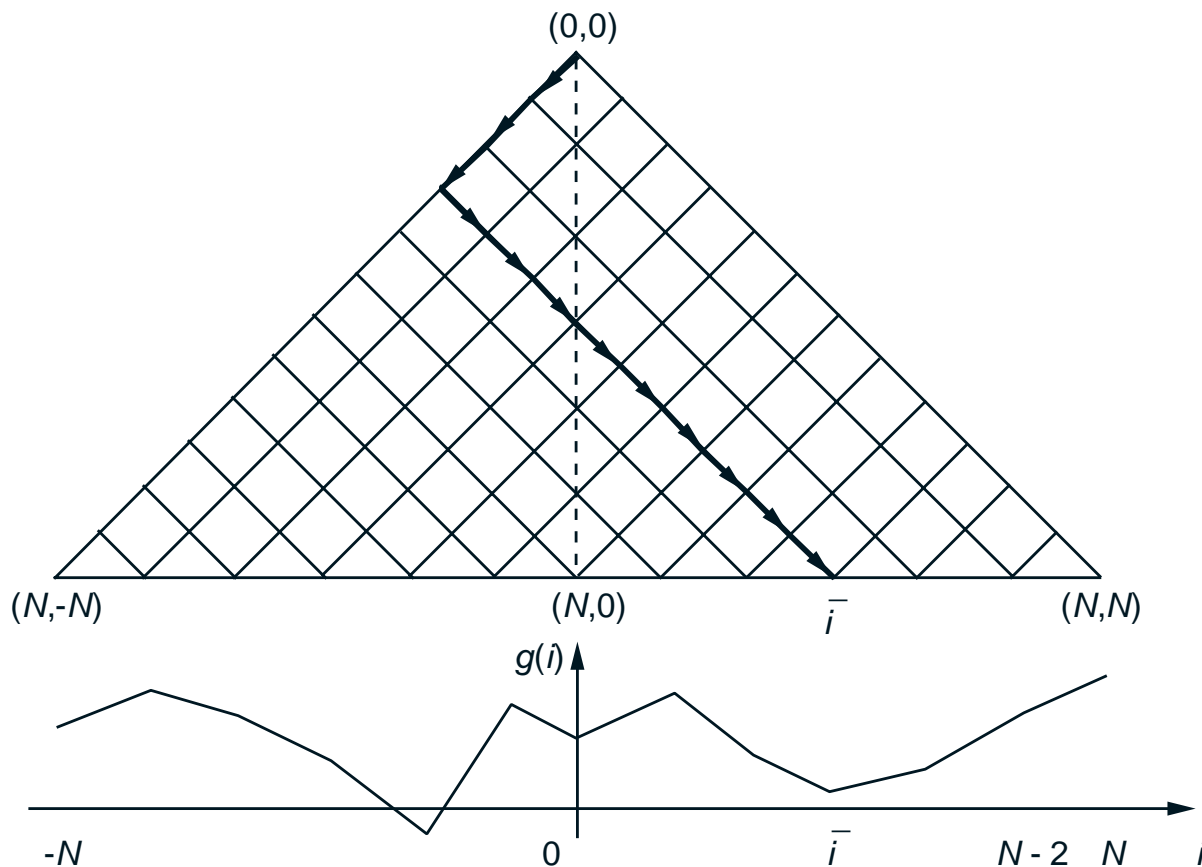
A CLASS OF GENERAL DISCRETE PROBLEMS

- Generic problem:
 - Given a graph with directed arcs
 - A special node s called the *origin*
 - A set of terminal nodes, called *destinations*, and a cost $g(i)$ for each destination i .
 - Find min cost path starting at the origin, ending at one of the destination nodes.
- Base heuristic: For any nondestination node i , constructs a path $(i, i_1, \dots, i_m, \bar{i})$ starting at i and ending at one of the destination nodes \bar{i} . We call \bar{i} the *projection* of i , and we denote $H(i) = g(\bar{i})$.
- Rollout algorithm: Start at the origin; choose the successor node with least cost projection



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*.

SEQUENTIAL CONSISTENCY

- The base heuristic is *sequentially consistent* if for every node i , whenever it generates the path $(i, i_1, \dots, i_m, \bar{i})$ starting at i , it also generates the path $(i_1, \dots, i_m, \bar{i})$ starting at the node i_1 (i.e., all nodes of its path have the same projection).
- Prime example of a sequentially consistent heuristic is a *greedy algorithm*. It uses an *estimate* $F(i)$ of the optimal cost starting from i .
- At the typical step, given a path (i, i_1, \dots, i_m) , where i_m is not a destination, the algorithm adds to the path a node i_{m+1} such that

$$i_{m+1} = \arg \min_{j \in N(i_m)} F(j)$$

- If the base heuristic is sequentially consistent, the cost of the rollout algorithm is no more than the cost of the base heuristic. In particular, if $(s, i_1, \dots, i_{\bar{m}})$ is the rollout path, we have

$$H(s) \geq H(i_1) \geq \dots \geq H(i_{\bar{m}-1}) \geq H(i_{\bar{m}})$$

where $H(i) = \text{cost of the heuristic starting from } i$.

SEQUENTIAL IMPROVEMENT

- We say that the base heuristic is *sequentially improving* if for every non-destination node i , we have

$$H(i) \geq \min_{j \text{ is neighbor of } i} H(j)$$

- If the base heuristic is sequentially improving, the cost of the rollout algorithm is no more than the cost of the base heuristic, starting from any node.
- Fortified rollout algorithm:
 - Simple variant of the rollout algorithm, where we keep the best path found so far through the application of the base heuristic.
 - If the rollout path deviates from the best path found, then follow the best path.
 - Can be shown to be a rollout algorithm with sequentially improving base heuristic for a slightly modified variant of the original problem.
 - Has the cost improvement property.

6.231 DYNAMIC PROGRAMMING

LECTURE 16

LECTURE OUTLINE

- More on rollout algorithms
- Simulation-based methods
- Approximations of rollout algorithms
- Rolling horizon approximations
- Discretization issues
- Other suboptimal approaches

ROLLOUT ALGORITHMS

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

$$\min_{u_k \in U_k(x_k)} Q_k(x_k, u_k),$$

where

$$Q_k(x_k, u_k) = E \left\{ g_k(x_k, u_k, w_k) + H_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

and $H_{k+1}(x_{k+1})$ is the cost-to-go of the heuristic.

- $Q_k(x_k, u_k)$ is called the *Q-factor* of (x_k, u_k) , and for a stochastic problem, its computation may involve Monte Carlo simulation.
- **Potential difficulty**: To minimize over u_k the *Q-factor*, we must form *Q-factor* differences $Q_k(x_k, u) - Q_k(x_k, \bar{u})$. This differencing often amplifies the simulation error in the calculation of the *Q-factors*.
- **Potential remedy**: Compare any two controls u and \bar{u} by simulating $Q_k(x_k, u) - Q_k(x_k, \bar{u})$ directly.

Q-FACTOR APPROXIMATION

- Here, instead of simulating the Q -factors, we approximate the costs-to-go $H_{k+1}(x_{k+1})$.
- Certainty equivalence approach: Given x_k , fix the future disturbances at “typical” values $\bar{w}_k, \dots, \bar{w}_{N-1}$ and approximate the Q -factors with

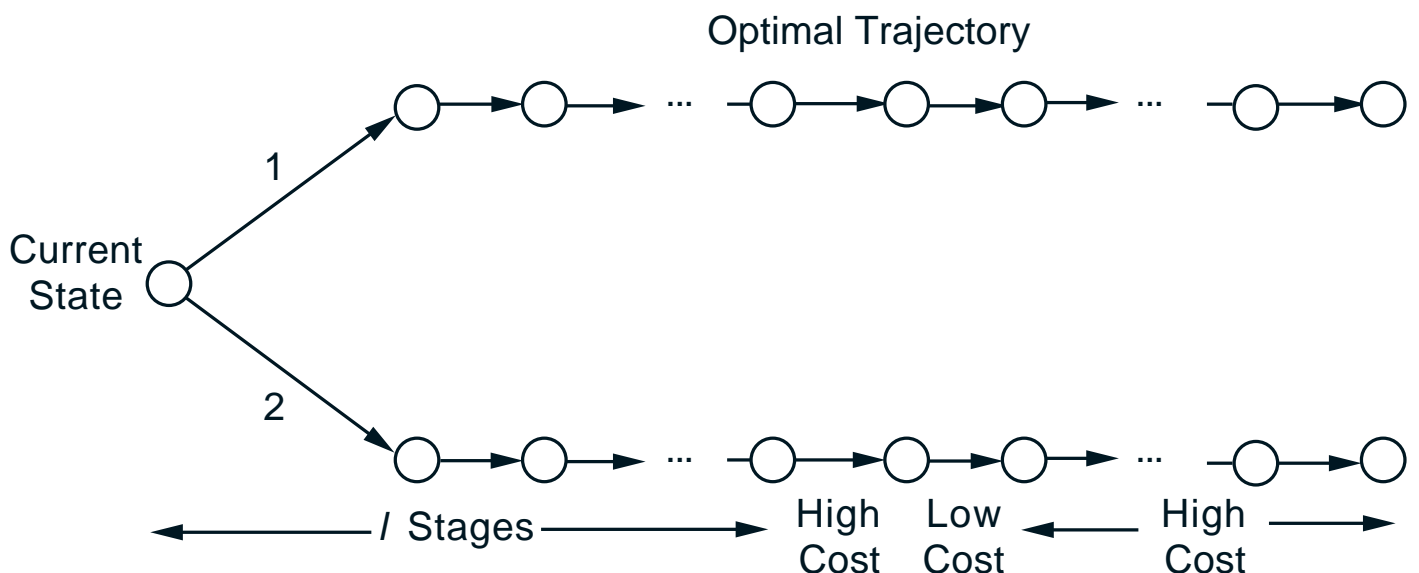
$$\tilde{Q}_k(x_k, u_k) = E \left\{ g_k(x_k, u_k, w_k) + \tilde{H}_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

where $\tilde{H}_{k+1}(f_k(x_k, u_k, w_k))$ is the cost of the heuristic with the disturbances fixed at the typical values.

- This is approximation of $H_{k+1}(f_k(x_k, u_k, w_k))$ by using a “single sample simulation.”
- Variant of the certainty equivalence approach: Approximate $H_{k+1}(f_k(x_k, u_k, w_k))$ by simulation using a small number of “representative samples” (scenarios).
- Alternative: Calculate (exact or approximate) values for the cost-to-go of the base policy at a limited set of state-time pairs, and then approximate H_{k+1} using an approximation architecture and a “least-squares fit.”

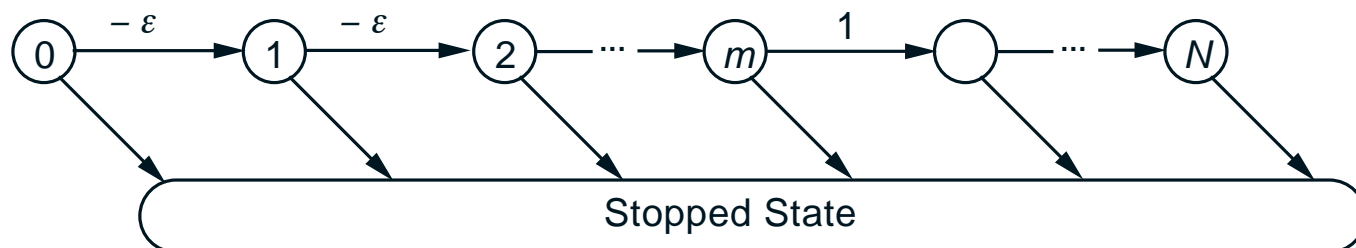
ROLLING HORIZON APPROACH

- This is an l -step lookahead policy where the cost-to-go approximation is just 0.
- Alternatively, the cost-to-go approximation is the terminal cost function g_N .
- A short rolling horizon saves computation.
- “Paradox”: It is not true that a longer rolling horizon always improves performance.
- Example: At the initial state, there are two controls available (1 and 2). At every other state, there is only one control.



ROLLING HORIZON COMBINED 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 $0 < \epsilon < 1/N$, 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!

DISCRETIZATION

- If the state space and/or control space is continuous/infinite, it must be replaced by a finite discretization.
- 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.
- Pitfalls with discretizing continuous time.
- The control constraint set changes a lot as we pass to the discrete-time approximation.
- Example:

$$\dot{x}_1(t) = u_1(t), \quad \dot{x}_2(t) = u_2(t),$$

with the control constraint $u_i(t) \in \{-1, 1\}$ for $i = 1, 2$. Compare with the discretized version

$$x_1(t+\Delta t) = x_1(t) + \Delta t u_1(t), \quad x_2(t+\Delta t) = x_2(t) + \Delta t u_2(t),$$

with $u_i(t) \in \{-1, 1\}$.

- “Convexification effect” of continuous time.

OTHER SUBOPTIMAL CONTROL APPROACHES

- Approximate the optimal cost-to-go functions $J_k(x_k)$ with functions $\tilde{J}_k(x_k, r_k)$, where r_k is a vector of unknown parameters, chosen to minimize some form of error in the DP equations.
- Approximate directly 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_0, \dots, s_{N-1} .

6.231 DYNAMIC PROGRAMMING

LECTURE 17

LECTURE OUTLINE

- Infinite horizon problems
- Stochastic shortest path problems
- Bellman's equation
- Dynamic programming – value iteration
- Examples

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 ($\alpha = 1$)
 - Discounted problems ($\alpha < 1$) with bounded cost per stage
 - Discounted and undiscounted problems with unbounded cost per stage
- Average cost problems

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

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 $J_0(x) \equiv 0$

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

- Typical results for total cost problems:

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

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

(Bellman's Equation). If $\mu(x)$ minimizes in Bellman's Eq., the policy $\{\mu, \mu, \dots\}$ is optimal.

- Bellman's Eq. always holds. 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 $J_\pi(x) = J^*(x)$ for all x .
- Special notation: For stationary policies $\pi = \{\mu, \mu, \dots\}$, we use $J_\mu(i)$ in place of $J_\pi(i)$.
- Assumption: 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

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

FINITENESS OF POLICY COST-TO-GO FUNCTIONS

- Let

$$\rho = \max_{\pi} \rho_{\pi}.$$

Note that ρ_{π} depends only on the first m components of the policy π , so that $\rho < 1$.

- 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:

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

- 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 ∞ .

EXAMPLE I

- 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 equations

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

EXAMPLE II

- A spider and a fly move along a straight line.
- The fly moves one unit to the left with probability p , one unit to the right with probability p , and stays where it is with probability $1 - 2p$.
- The spider moves one unit towards the fly if its distance from the fly is more than one unit.
- If the spider is one unit away from the fly, it will either move one unit towards the fly or stay where it is.
- If the spider and the fly land in the same position, the spider captures the fly.
- The spider's objective is to capture the fly in minimum expected time.
- This is an SSP w/ state = the distance between spider and fly ($i = 1, \dots, n$ and $t = 0$ the termination state).
- There is control choice only at state 1.

EXAMPLE II (CONTINUED)

- For $M = \text{move}$, and $\bar{M} = \text{don't move}$

$$p_{11}(M) = 2p, \quad p_{10}(M) = 1 - 2p,$$

$$p_{12}(\bar{M}) = p, \quad p_{11}(\bar{M}) = 1 - 2p, \quad p_{10}(\bar{M}) = p,$$

$$p_{ii} = p, \quad p_{i(i-1)} = 1 - 2p, \quad p_{i(i-2)} = p, \quad i \geq 2,$$

with all other transition probabilities being 0.

- Bellman's equation:

$$J^*(i) = 1 + pJ^*(i) + (1 - 2p)J^*(i-1) + pJ^*(i-2), \quad i \geq 2$$

$$J^*(1) = 1 + \min[2pJ^*(1), pJ^*(2) + (1 - 2p)J^*(1)]$$

w/ $J^*(0) = 0$. Substituting $J^*(2)$ in Eq. for $J^*(1)$,

$$J^*(1) = 1 + \min \left[2pJ^*(1), \frac{p}{1-p} + \frac{(1-2p)J^*(1)}{1-p} \right].$$

- Work from here to find that when one unit away from the fly it is optimal *not to move if and only if* $p \geq 1/3$.

6.231 DYNAMIC PROGRAMMING

LECTURE 18

LECTURE OUTLINE

- Stochastic shortest path problems
- Policy iteration
- Linear programming
- 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 $J_\pi(x) = J^*(x)$ for all x .
- Special notation: For stationary policies $\pi = \{\mu, \mu, \dots\}$, we use $J_\mu(i)$ in place of $J_\pi(i)$.
- Assumption: 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 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:

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

- 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 ∞ .

EXAMPLE

- A spider and a fly move along a straight line.
- The fly moves one unit to the left with probability p , one unit to the right with probability p , and stays where it is with probability $1 - 2p$.
- The spider moves one unit towards the fly if its distance from the fly is more than one unit.
- If the spider is one unit away from the fly, it will either move one unit towards the fly or stay where it is.
- If the spider and the fly land in the same position, the spider captures the fly.
- The spider's objective is to capture the fly in minimum expected time.
- This is an SSP w/ state = the distance between spider and fly ($i = 1, \dots, n$ and $t = 0$ the termination state).
- There is control choice only at state 1.

EXAMPLE (CONTINUED)

- For $M = \text{move}$, and $\bar{M} = \text{don't move}$

$$p_{11}(M) = 2p, \quad p_{10}(M) = 1 - 2p,$$

$$p_{12}(\bar{M}) = p, \quad p_{11}(\bar{M}) = 1 - 2p, \quad p_{10}(\bar{M}) = p,$$

$$p_{ii} = p, \quad p_{i(i-1)} = 1 - 2p, \quad p_{i(i-2)} = p, \quad i \geq 2,$$

with all other transition probabilities being 0.

- Bellman's equation:

$$J^*(i) = 1 + pJ^*(i) + (1 - 2p)J^*(i-1) + pJ^*(i-2), \quad i \geq 2$$

$$J^*(1) = 1 + \min[2pJ^*(1), pJ^*(2) + (1 - 2p)J^*(1)]$$

w/ $J^*(0) = 0$. Substituting $J^*(2)$ in Eq. for $J^*(1)$,

$$J^*(1) = 1 + \min \left[2pJ^*(1), \frac{p}{1-p} + \frac{(1-2p)J^*(1)}{1-p} \right].$$

- Work from here to find that when one unit away from the fly it is optimal *not to move if and only if* $p \geq 1/3$.

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

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

- For each k , $J_{\mu^{k+1}}(i) \leq J_{\mu^k}(i)$ for all i , with strict inequality for at least one i if μ^k is not optimal.
- A policy cannot be repeated, and since there are finitely many stationary policies, an optimal policy is obtained in a finite number of iterations.

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 $u \in U(i)$.

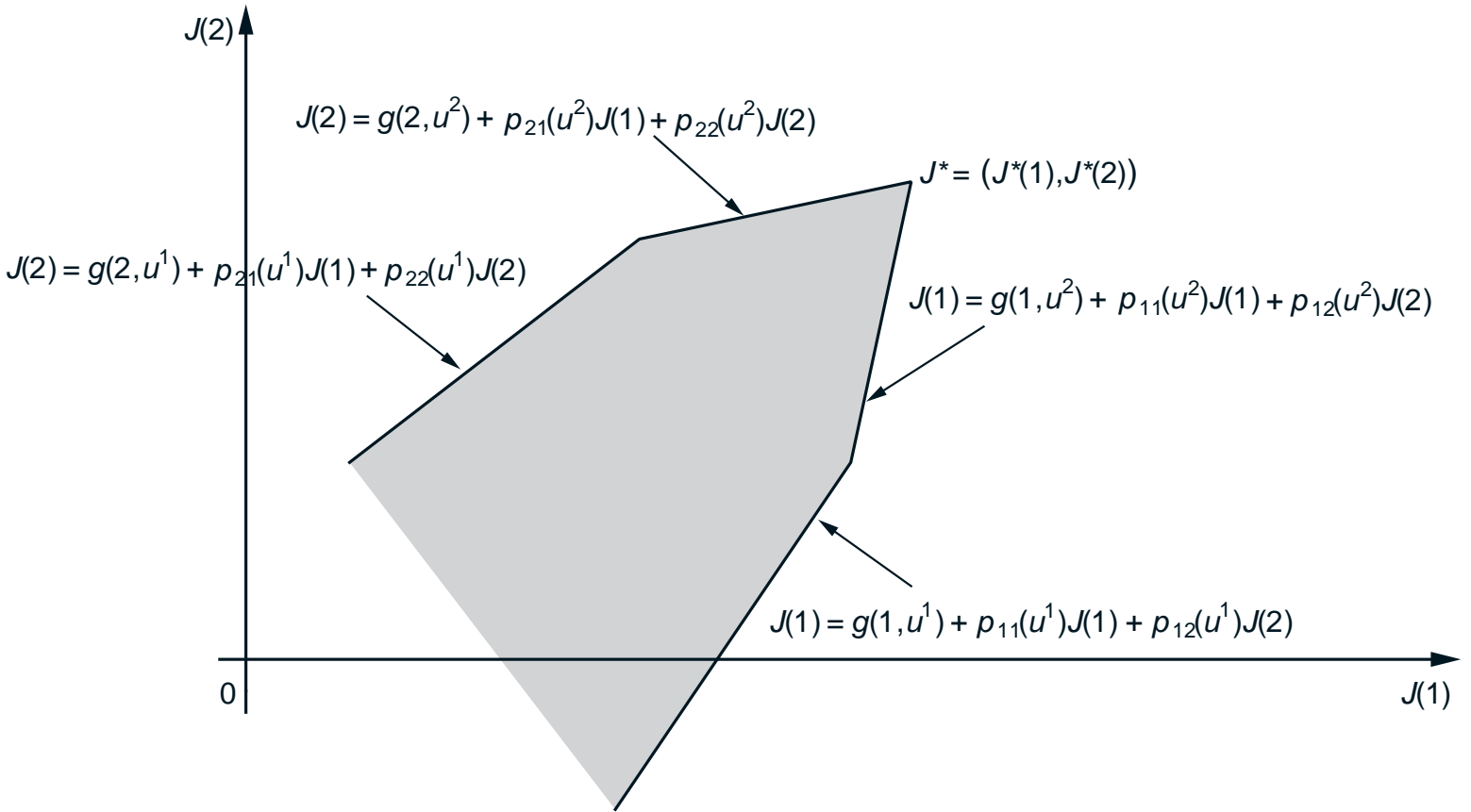
- Proof: If we use value iteration to generate a sequence of vectors $J_k = (J_k(1), \dots, J_k(n))$ starting with a J_0 such that

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

Then, $J_k(i) \leq J_{k+1}(i)$ for all k and i (monotonicity 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).

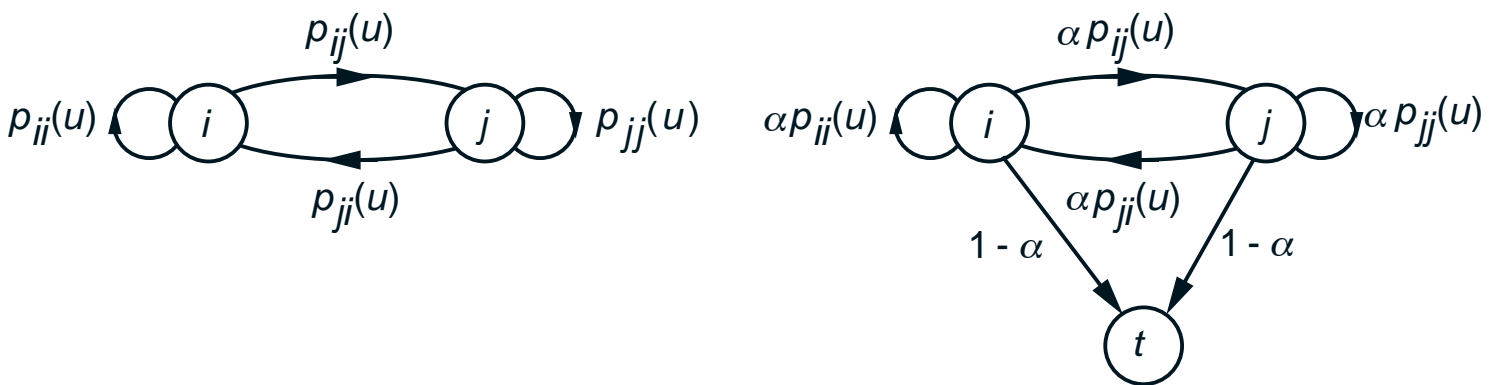
LINEAR PROGRAMMING (CONTINUED)



- Drawback: For large n the dimension of this program is very large.

DISCOUNTED PROBLEMS

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



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

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

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

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

DISCOUNTED PROBLEMS (CONTINUED)

- Policy iteration converges finitely to an optimal, 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

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

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 19

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

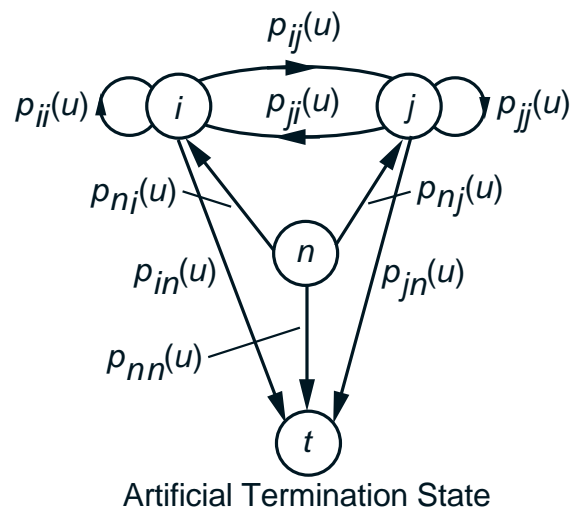
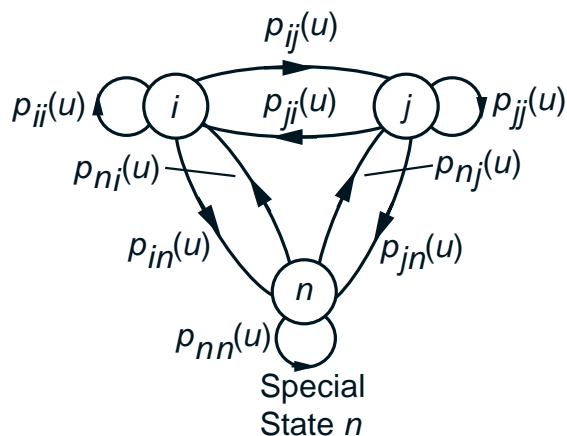
- Stationary system with finite number of states and controls
- Minimize over policies $\pi = \{\mu_0, \mu_1, \dots\}$

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

- Important characteristics (not shared by other types of infinite horizon problems)
 - For any fixed N , the cost incurred up to time N does not matter (only the state that we are at time N matters)
 - If all states “communicate” the optimal cost is independent of the 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.

CONNECTION WITH SSP

- Assumption: State n is such that for some integer $m > 0$, and for all initial states and all policies, n is visited with positive probability at least once within the first m stages.
- 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 stage cost at i of the SSP problem be

$$g(i, u) - \lambda^*, \quad \lambda^*: \text{optimal av. cost per stage from } n$$

CONNECTION WITH SSP (CONTINUED)

- View the average cost problem as a *minimum cycle cost problem*: Find a stationary policy μ that minimizes the *expected cost per cycle*

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

where for a fixed μ ,

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

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

- Justification: Intuitively, $\lambda^* = \text{optimal cycle cost}$, so

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

with equality if μ is optimal.

- Thus, the optimal μ must minimize over μ the expression $C_{nn}(\mu) - N_{nn}(\mu)\lambda^*$, which is the expected cost of μ starting from n in the SSP with stage costs $g(i, u) - \lambda^*$.

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

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

- If $\mu^*(i)$ attains the min for each i , μ^* is optimal.

MORE ON THE CONNECTION W/ SSP

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

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

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

- Note that we don't know λ^* , so we cannot 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^* .

MORE ON 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 $h_k \rightarrow h^*$ (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*: 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

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

The algorithm terminates with an optimal policy.

6.231 DYNAMIC PROGRAMMING

LECTURE 20

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 ($t_0 = 0$)
- $x_k = x(t_k)$: We have $x(t) = x_k$ for $t_k \leq t < t_{k+1}$.
- $u_k = u(t_k)$: We have $u(t) = u_k$ for $t_k \leq t < t_{k+1}$.
- In place of transition probabilities, we have *transition distributions*

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

- 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 conditional distribution function (CDF) of τ given i, j, u is

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

for all i, j, u such that $p_{ij}(u) > 0$.

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 interval between the transition to state i and the transition to the next state is exponentially distributed with parameter $\nu_i(u)$; that is,

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

- The exponential distribution is *memoryless*. This implies that for a given policy, the system is a continuous-time Markov chain.

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 $\beta > 0$)

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

- Average cost per unit time

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

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

DISCOUNTED PROBLEMS – COST CALCULATION

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

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

where $J_{\pi_1}(j)$ is the cost-to-go of the policy $\pi_1 = \{\mu_1, \mu_2, \dots\}$

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

$$\begin{aligned} G(i, u) &= E_j \{ E_\tau \{ \text{transition cost} \mid j \} \} \\ &= \sum_{\{j \mid p_{ij}(u) > 0\}}^n p_{ij}(u) \int_0^\infty \left(\int_0^\tau e^{-\beta t} g(i, u) dt \right) \frac{dQ_{ij}(\tau, u)}{p_{ij}(u)} \\ &= \sum_{j=1}^n \int_0^\infty \frac{1 - e^{-\beta\tau}}{\beta} g(i, u) dQ_{ij}(\tau, u) \end{aligned}$$

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

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

COST CALCULATION (CONTINUED)

- Also

$$\begin{aligned} E\{e^{-\beta\tau} J_{\pi_1}(j)\} &= E_j\{e^{-\beta\tau} J_{\pi_1}(j)\} \\ &= E_j\{E\{e^{-\beta\tau} \mid j\} J_{\pi_1}(j)\} \\ &= \sum_{\{j \mid p_{ij}(u) > 0\}}^n p_{ij}(u) \left(\int_0^\infty e^{-\beta\tau} \frac{dQ_{ij}(\tau, u)}{p_{ij}(u)} J_{\pi_1}(j) \right) \\ &= \sum_{j=1}^n m_{ij}(\mu(i)) J_{\pi_1}(j) \end{aligned}$$

where $m_{ij}(u)$ is the “effective discount factor”

$$\begin{aligned} m_{ij}(u) &= \int_0^\infty e^{-\beta\tau} dQ_{ij}(\tau, u) \\ &= p_{ij}(u) E\{\text{discount factor} \mid j\} \end{aligned}$$

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

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

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

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

- 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 $[0, \tau_{\max}]$.
- 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

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

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:

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

AVERAGE COST MANUFACTURER'S EXAMPLE

- 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

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

and there is also the “instantaneous” cost

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

- Bellman's equation:

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 21

LECTURE OUTLINE

- We start a four-lecture sequence on advanced dynamic programming and neuro-dynamic programming topics
- Discounted problems - infinite state space
- Mappings underlying infinite horizon DP
- Geometric interpretation of value and policy iteration
- Approximate policy iteration

DISCOUNTED PROBLEMS W/ BOUNDED COST

- System

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

- 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 $g(x, u, w)$: bounded over (x, u, w) , and $\alpha < 1$.

- 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

- 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 steps: 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

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

THE THREE PROPERTIES

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

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

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

- **Additivity 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 [$e(x) \equiv 1$].

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

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

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

“SHORTHAND” ANALYSIS

- **Contraction mapping theorem:** The contraction property implies that:
 - T has a unique fixed point, J^* , which is the limit of $T^k J$ for any (bounded) J .
 - For each μ , T_μ has a unique fixed point, J_μ , which is the limit of $T_\mu^k J$ for any J .
- **Convergence rate:** For all k ,

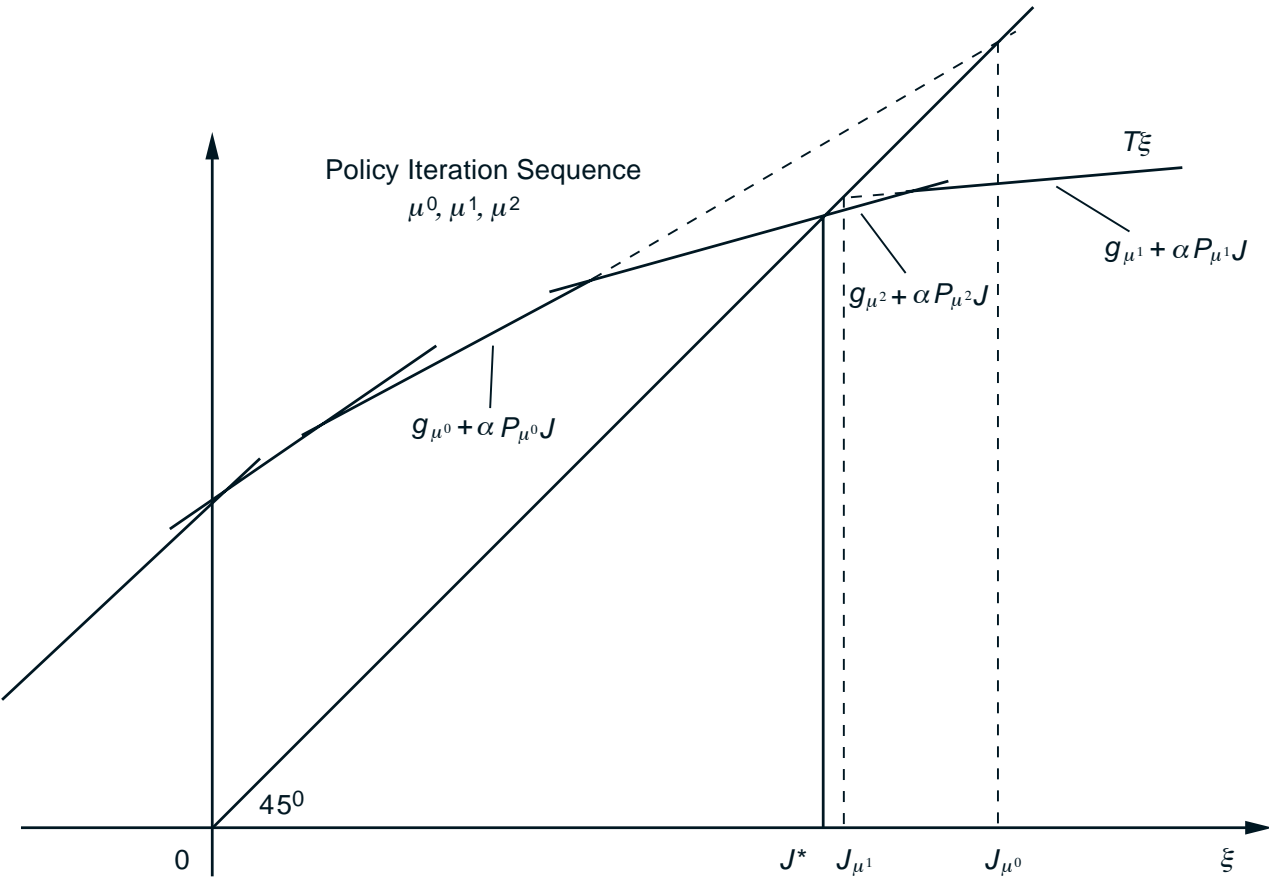
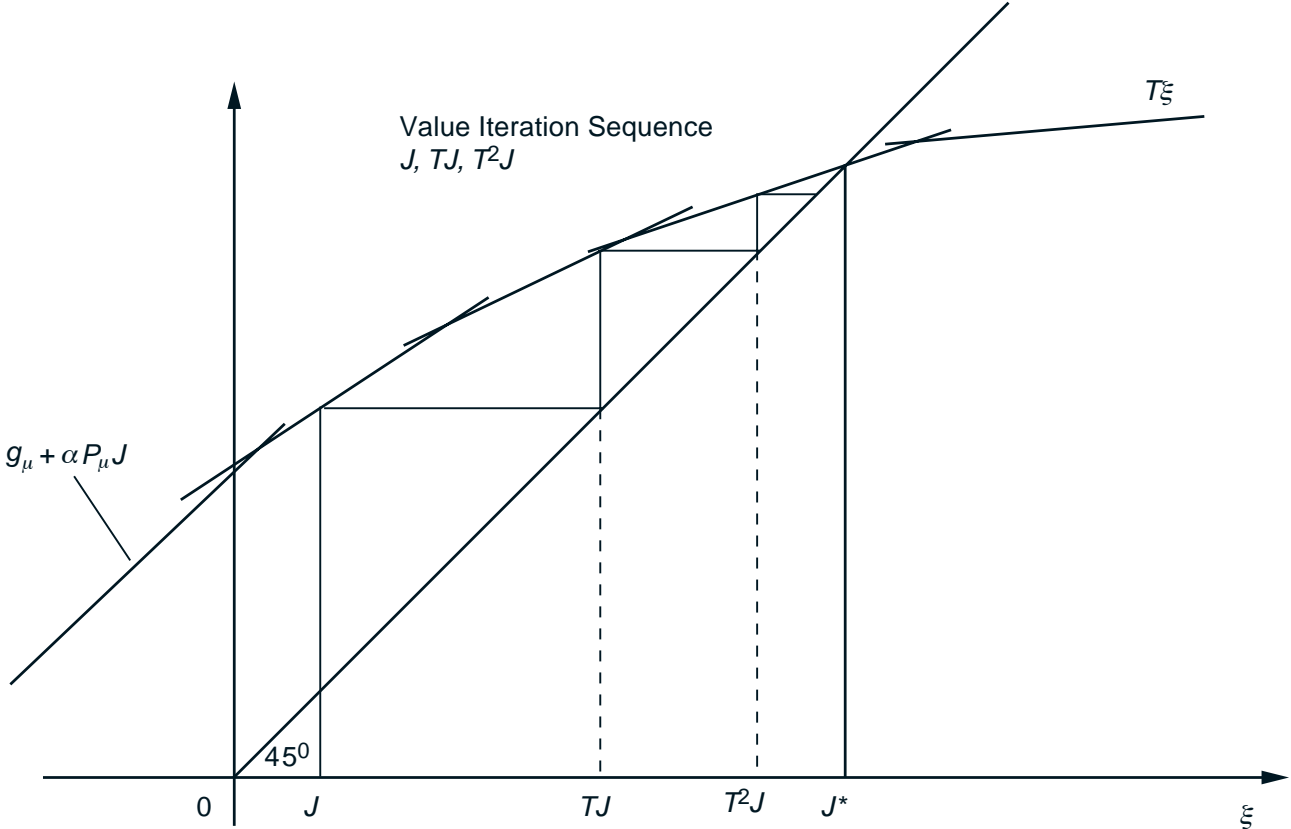
$$\max_x |(T^k J)(x) - J^*(x)| \leq \alpha^k \max_x |J(x) - J^*(x)|$$

- An assortment of other analytical and computational results are based on the contraction property, e.g, error bounds, computational enhancements, etc.
- **Example:** If we execute value iteration *approximately*, so we compute TJ within an ϵ -error, i.e.,

$$\max_x |\tilde{J}(x) - (TJ)(x)| \leq \epsilon,$$

in the limit we obtain J^* within an $\epsilon/(1 - \alpha)$ error.

GEOMETRIC INTERPRETATIONS



APPROXIMATE POLICY ITERATION

- 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 also 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 Bounds:** The sequence $\{\mu^k\}$ generated by the approximate policy iteration algorithm satisfies

$$\limsup_{k \rightarrow \infty} \max_{x \in S} (J_{\mu^k}(x) - J^*(x)) \leq \frac{\epsilon + 2\alpha\delta}{(1 - \alpha)^2}$$

- **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^* .

6.231 DYNAMIC PROGRAMMING

LECTURE 22

LECTURE OUTLINE

- Stochastic shortest problems in greater generality
- Proper and improper policies
- Counterexamples
- Approximate policy iteration
- Simulation-based policy iteration
- Actor-critic interpretation

UNDISCOUNTED PROBLEMS

- System

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

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

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

- Shorthand notation for DP mappings

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

- For any stationary policy μ

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

- Neither T nor T_μ are contractions in general. Some, but not all, of the nice theory holds, thanks to the monotonicity of T and T_μ .

- Some of the nice theory is recovered in SSP problems because of the termination state.

STOCHASTIC SHORTEST PATH PROBLEMS I

- Assume: cost-free term. state t , a finite number of states $1, \dots, n$, and finite number of controls
- Modified mapping T and T_μ :

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

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

- **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 then T_μ is a contraction with respect to some weighted max norm

$$\max_i \frac{1}{v_i} |(T_\mu J)(i) - (T_\mu J')(i)| \leq \alpha \max_i \frac{1}{v_i} |J(i) - J'(i)|$$

- If all μ are proper, then T is similarly a contraction (this is the case discussed in Ch. 7).

STOCHASTIC SHORTEST PATH PROBLEMS II

- The theory can be pushed one step further. Assume that:
 - (a) There exists at least one proper policy
 - (b) For each improper μ , we have $T_\mu(i) = \infty$ for some state 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
 - States $\langle == \rangle$ nodes
 - Controls $\langle == \rangle$ arcs
 - Termination state $\langle == \rangle$ the destination
 - Assumption (a) $\langle == \rangle$ every node is connected to the destination
 - Assumption (b) $\langle == \rangle$ every cycle has positive cost

PATHOLOGIES: THE BLACKMAILER'S DILEMMA

- Two states, state 1 and the termination state t .
- At state 1, choose a control $u \in (0, 1]$ (the blackmail amount demanded), and move to t at no cost with probability u^2 , or stay in 1 at a cost $-u$ with probability $1 - u^2$.
- Every stationary policy is proper, but the control set is not finite.
- For any stationary μ with $\mu(1) = u$, we have

$$J_\mu(1) = -(1 - u^2)u + (1 - u^2)J_\mu(1)$$

from which $J_\mu(1) = -\frac{1-u^2}{u}$

- 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.)

APPROXIMATE POLICY ITERATION

- Suppose that the policy evaluation is approximate, according to,

$$\max_{i=1,\dots,n} |J_k(i) - J_{\mu^k}(i)| \leq \delta, \quad k = 0, 1, \dots$$

and policy improvement is also approximate, according to,

$$\max_{i=1,\dots,n} |(T_{\mu^{k+1}} J_k)(i) - (T J_k)(i)| \leq \epsilon, \quad k = 0, 1, \dots$$

where δ and ϵ are some positive scalars.

- Assume that all policies generated by the method are proper (they are guaranteed to be if $\delta = \epsilon = 0$, but not in general).

- **Error Bounds:** The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \rightarrow \infty} \max_{i=1,\dots,n} (J_{\mu^k}(i) - J^*(i)) \leq \frac{n(1 - \rho + n)(\epsilon + 2\delta)}{(1 - \rho)^2}$$

where $\rho = \max_{\substack{i=1,\dots,n \\ \mu: \text{proper}}} P\{x_n \neq t \mid x_0 = i, \mu\}$

SIMULATION-BASED POLICY EVALUATION

- Given μ , suppose we want to calculate J_μ by simulation.
- Generate by simulation sample costs. Approximation:

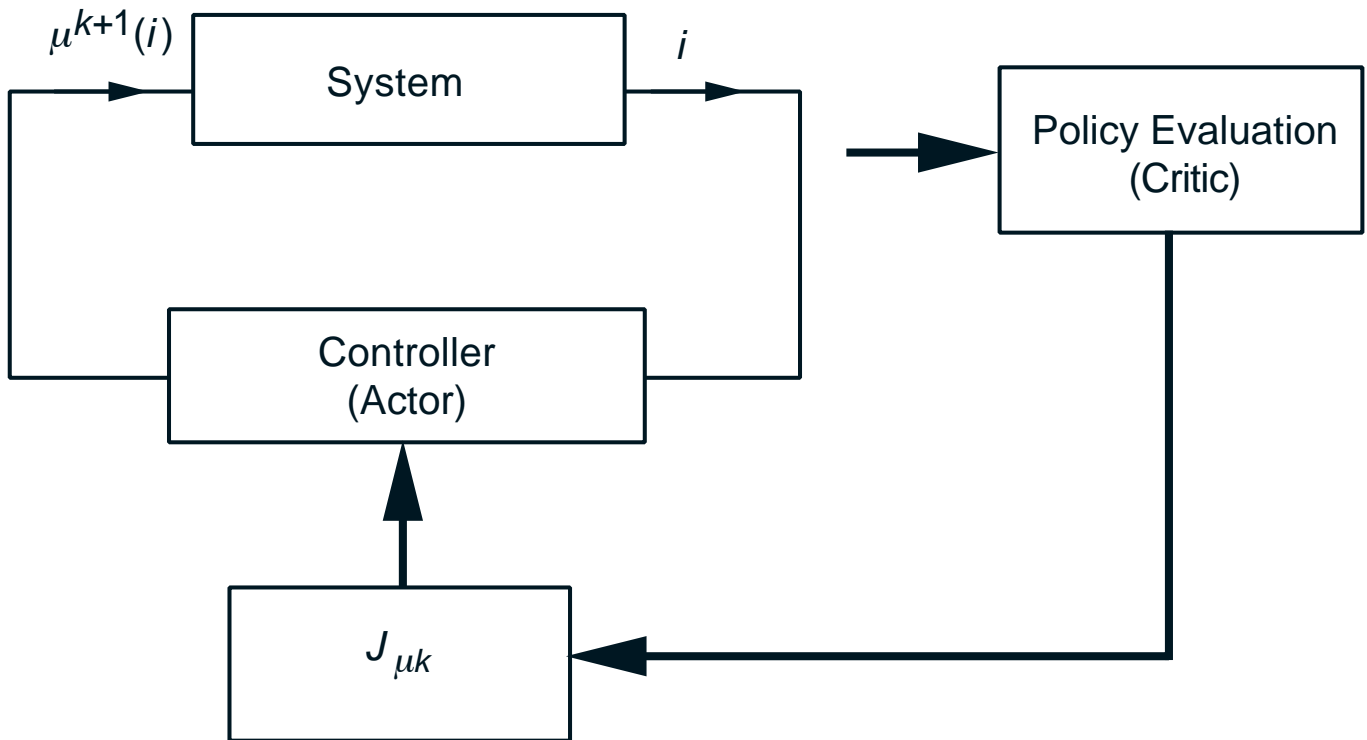
$$J_\mu(i) \approx \frac{1}{M_i} \sum_{m=1}^{M_i} c(i, m)$$

$c(i, m)$: m th sample cost starting from state i

- Approximating each $J_\mu(i)$ is impractical for a large state space. Instead, a “compact representation” $\tilde{J}_\mu(i, r)$ may be used, where r is a tunable parameter vector. We may calculate an optimal value r^* of r by a least squares fit

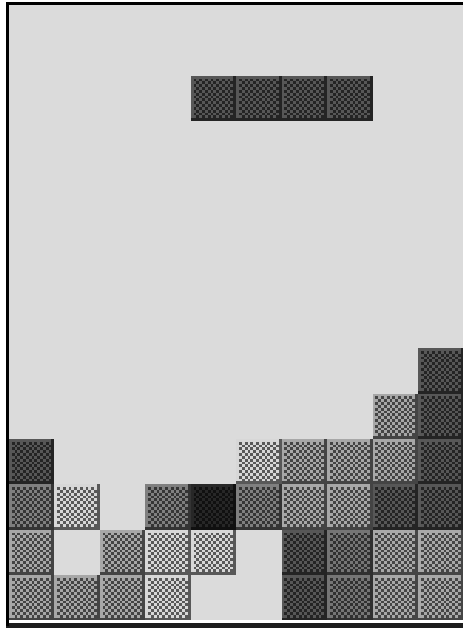
$$r^* = \arg \min_r \sum_{i=1}^n \sum_{m=1}^{M_i} |c(i, m) - \tilde{J}_\mu(i, r)|^2$$

ACTOR-CRITIC INTERPRETATION



- Policy evaluation may be approximate, e.g., using limited simulation

EXAMPLE: TETRIS



- The state consists of the board position i , and the shape of the current falling block (astronomically large number of states).
- Can be shown that all policies are proper!!
- Approximating function with feature extraction

$$\tilde{J}(i, r) = \sum_{k=1}^m r_k w_k(i),$$

where $r = (r_1, \dots, r_m)$ is the parameter vector and $w_k(i)$ is the k th feature.

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

- Overview of simulation-based methods for exact and approximate DP
- Lookup table methods
- Q -Learning
- Aggregation

SIMULATION-BASED METHODS (NDP)

- There are many proposals, but the methods for which there is solid theory are of two types:
 - (a) Policy evaluation methods, to be used as part of an exact or approximate *policy iteration* scheme.
 - The policy is fixed.
 - As a special case we obtain the rollout method (only one policy iteration is used).
 - The cost of the policy may be calculated in several different forms: (1) For all states (lookup table representation) or (2) Through an approximation architecture (compact representation) or (2) Through on-line simulation as needed (rollout algorithm).
 - (b) *Q*-Learning
 - The policy is not fixed.
 - Can be viewed as a form of an on-line simulation-based *value iteration* method.
 - The only available theory applies to the lookup table representation case

POLICY EVALUATION METHODS

- The policy is fixed.
- Many simulation trajectories are generated.
- The weight vector r of an approximation architecture $\tilde{J}(i, r)$ is adjusted using some kind of “least squares scheme” (off-line, or on-line as the simulation trajectories are generated).
- There is a large variety of on-line methods, [TD(λ), λ -policy iteration, etc].
- For on-line methods, a sequence $\{r^k\}$ of parameter vectors is generated.
 - There may be one or more transitions between parameter vector changes
 - There is solid theory only for linear approximation architectures (and under some technical assumptions)
- Typical result: In the limit, as the number of simulation-generated transitions goes to ∞ , the sequence of generated parameter vectors converges to a limit.

Q-LEARNING I

- To implement an optimal policy, what we need are the Q -factors defined for each pair (i, u) by

$$Q(i, u) = \sum_j p_{ij}(u) (g(i, u, j) + J^*(j))$$

- Bellman's equation is $J^*(j) = \min_{u' \in U(j)} Q(j, u')$, so the Q -factors solve the system of equations

$$Q(i, u) = \sum_j p_{ij}(u) \left(g(i, u, j) + \min_{u' \in U(j)} Q(j, u') \right), \forall (i, u)$$

- One possibility is to solve this system iteratively by a form of value iteration

$$Q(i, u) := (1 - \gamma)Q(i, u) + \gamma \sum_j p_{ij}(u) \left(g(i, u, j) + \min_{u' \in U(j)} Q(j, u') \right),$$

where γ is a stepsize parameter with $\gamma \in (0, 1]$, that may change from one iteration to the next.

Q-LEARNING II

- The *Q-learning method* is an approximate version of this iteration, whereby the expected value is replaced by a single sample, i.e.,

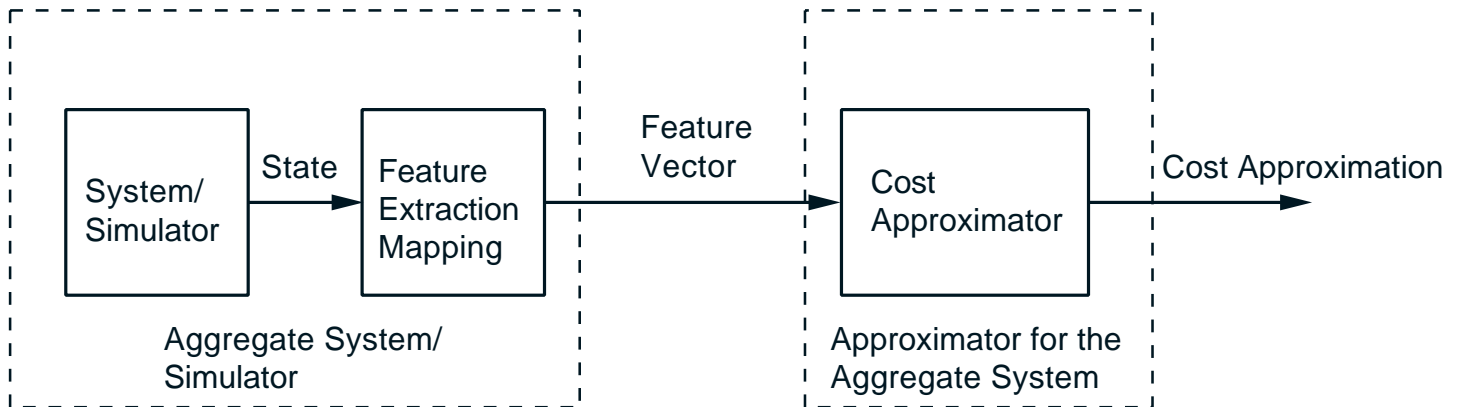
$$Q(i, u) := Q(i, u) + \gamma \left(g(i, u, j) + \min_{u' \in U(j)} Q(j, u') - Q(i, u) \right)$$

- Here j and $g(i, u, j)$ are generated from the pair (i, u) by simulation, i.e., according to the transition probabilities $p_{ij}(u)$.
- Thus Q-learning can be viewed as a combination of value iteration and simulation.
- Convergence of the method to the (optimal) Q factors can be shown under some reasonable (but quite technical) assumptions.
- Strong connections with the theory of stochastic iterative algorithms (such as stochastic gradient methods).
- Challenging analysis, limited practicality (only for a small number of states).

AGGREGATION I

- Another major idea in approximate DP is to approximate the cost-to-go function of the problem with the cost-to-go function of a simpler problem.
- Idea of aggregation approach:
 - Lump many states together into a few “aggregate” states
 - View the aggregate states as the states of an “aggregate” system
 - Formulate and solve (optimally) the “aggregate” problem by any kind of value or policy iteration method (including simulation-based methods, such as Q -learning)
 - Use the optimal cost of the aggregate problem for a piecewise-constant approximation of the optimal cost of the original problem (all states that belong to the same aggregate state are restricted to have the same cost, the optimal cost of the aggregate state)
- The aggregate problem could also be solved approximately

AGGREGATION II



- Main steps to define the aggregate system
- Form the aggregate states by partitioning the original state space (features can be used for this).
 - Each aggregate state is a subset S of states of the original system
 - Each state of the original system belongs to a unique aggregate state.
- Define the dynamics of the aggregate system

Current aggregate state $S \mapsto$ New aggregate state S'

Example: If the current aggregate state is S , generate a “typical” state i within S in some probabilistic way, then generate j according to the p_{ij} , then declare S' to be the aggregate state to which j belongs.