LECTURE SLIDES ON DYNAMIC PROGRAMMING

BASED ON LECTURES GIVEN AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASS

FALL 2004

DIMITRI P. BERTSEKAS

These lecture slides are based on the book: "Dynamic Programming and Optimal Control: 2nd edition," Vols. I and II, Athena Scientific, 2002, by Dimitri P. Bertsekas; see

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

Last Updated: December 2004

LECTURE CONTENTS

- These slides consist of 24 Lectures, whose summary is given in the next 24 slides
- Lectures 1-2: Basic dynamic programming algorithm (Chapter 1)
- Lectures 3-4: Deterministic discrete-time and shortest path problems (Chapter 2)
- Lectures 5-6: Stochastic discrete-time problems (Chapter 4)
- Lectures 7-9: Deterministic continuous-time optimal control (Chapter 4)
- Lectures 10-12: Problems of imperfect state information (Chapter 5)
- Lectures 13-16: Approximate DP - suboptimal control (Chapter 6)
- Lectures 17-20: Introduction to infinite horizon problems (Chapter 7)
- Lectures 21-24: Advanced topics on infinite horizon and approximate DP (Volume II)

6.231 DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

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

6.231 DYNAMIC PROGRAMMING

LECTURE 2

LECTURE OUTLINE

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

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 5

LECTURE OUTLINE

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

6.231 DYNAMIC PROGRAMMING

LECTURE 6

LECTURE OUTLINE

- Stopping problems
- Scheduling problems
- Other applications

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 as a continuous-time limit of the DP algorithm
- The Hamilton-Jacobi-Bellman equation as a sufficient condition
- Examples

6.231 DYNAMIC PROGRAMMING

LECTURE 8

LECTURE OUTLINE

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 10

LECTURE OUTLINE

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

6.231 DYNAMIC PROGRAMMING

LECTURE 11

LECTURE OUTLINE

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 13

LECTURE OUTLINE

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 15

LECTURE OUTLINE

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 17

LECTURE OUTLINE

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

6.231 DYNAMIC PROGRAMMING

LECTURE 18

LECTURE OUTLINE

- Stochastic shortest path problems
- Policy iteration
- Linear programming
- Discounted problems

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 21

LECTURE OUTLINE

- With this lecture, we start a four-lecture sequence on advanced dynamic programming and neuro-dynamic programming topics. References:
  - Dynamic Programming and Optimal Control, Vol. II, by D. Bertsekas
  - Neuro-Dynamic Programming, by D. Bertsekas and J. Tsitsiklis
- **1st Lecture:** Discounted problems with infinite state space, stochastic shortest path problem
- **2nd Lecture:** DP with cost function approximation
- **3rd Lecture:** Simulation-based policy and value iteration, temporal difference methods
- **4th Lecture:** Other approximation methods: Q-learning, state aggregation, approximate linear programming, approximation in policy space

6.231 DYNAMIC PROGRAMMING

LECTURE 22

LECTURE OUTLINE

- Approximate DP for large/intractable problems
- Approximate policy iteration
- Simulation-based policy iteration
- Actor-critic interpretation
- Learning how to play tetris: A case study
- Approximate value iteration with function approximation

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

- Simulation-based policy and value iteration methods
- λ-Least Squares Policy Evaluation method
- Temporal differences implementation
- Policy evaluation by approximate value iteration
- TD(λ)

6.231 DYNAMIC PROGRAMMING

LECTURE 24

LECTURE OUTLINE

- Additional methods for approximate DP
- Q-Learning
- Aggregation
- Linear programming with function approximation
- Gradient-based approximation in policy space

6.231 DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

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

DP AS AN OPTIMIZATION METHODOLOGY

- Basic optimization problem

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

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

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

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

    where w is a random parameter.

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

BASIC STRUCTURE OF STOCHASTIC DP

- Discrete-time system

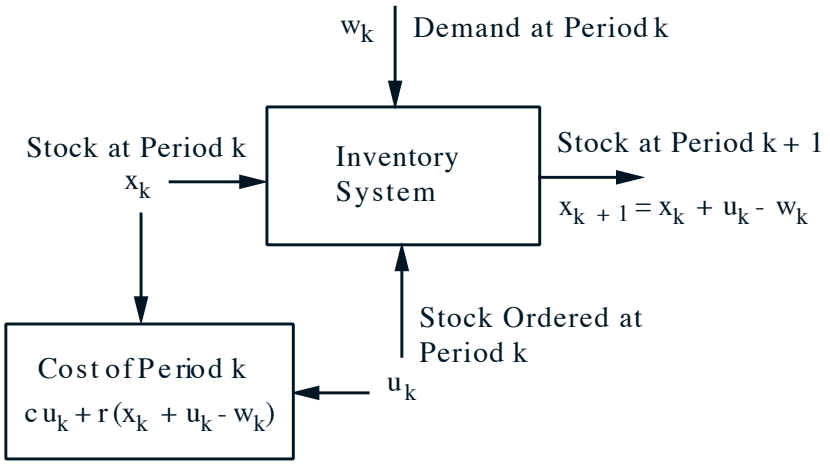
$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1$$

  - 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

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

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

- 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 $w_{k-1}, \ldots, w_0$, 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

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

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

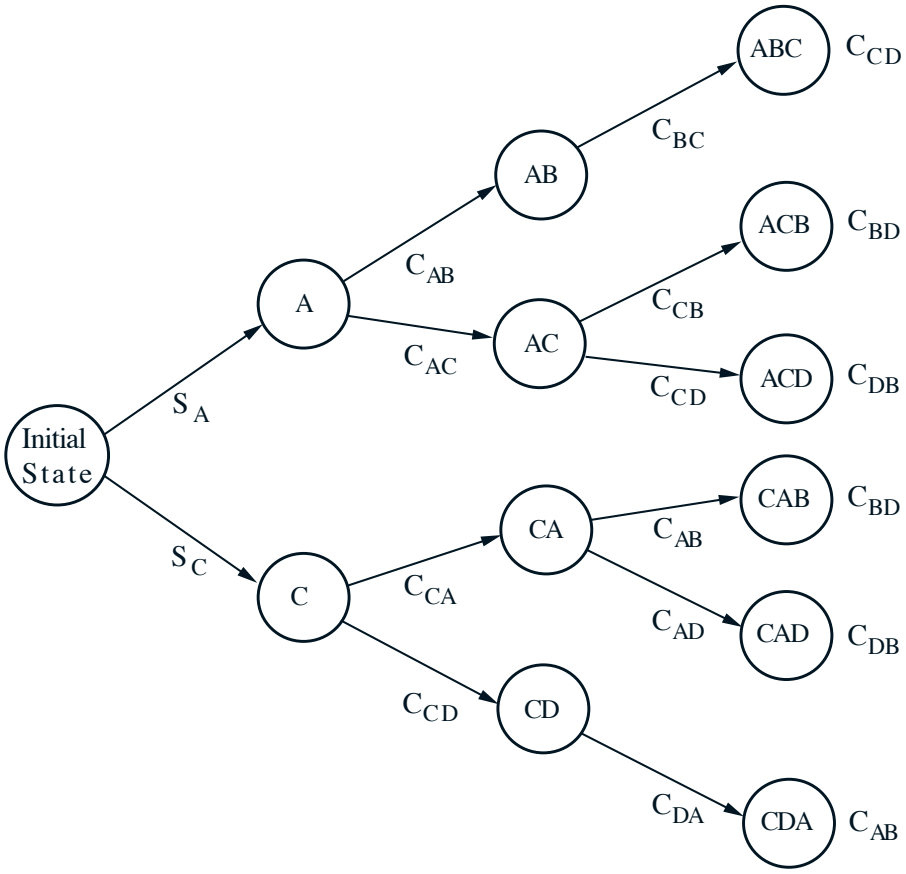
$$u_k \in U(x_k)$$

  - w_k is random and generated according to a distribution

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

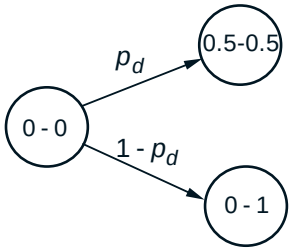
DETERMINISTIC FINITE-STATE PROBLEMS

- 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

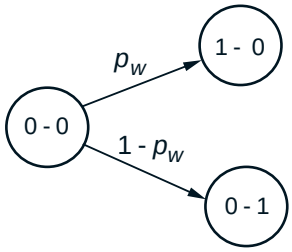


STOCHASTIC FINITE-STATE PROBLEMS

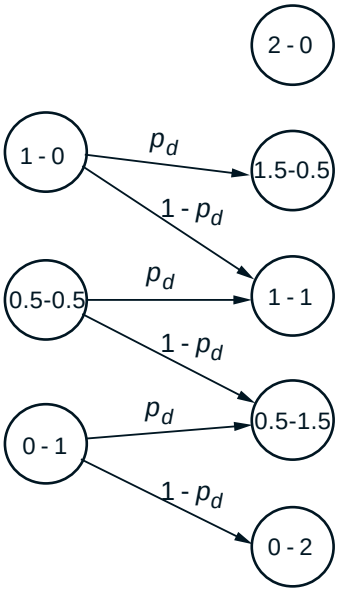
- Example: Find two-game chess match strategy
- *Timid* play draws with prob. $p_d > 0$ and loses with prob. $1 - p_d$. *Bold* play wins with prob. $p_w < 1/2$ and loses with prob. $1 - p_w$



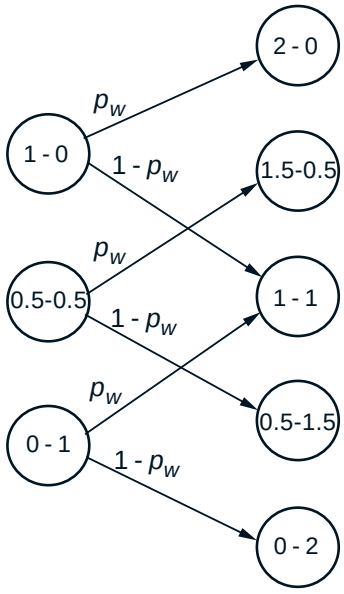
1st Game / Timid Play



1st Game / Bold Play



2nd Game / Timid Play



2nd Game / Bold Play

BASIC PROBLEM

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

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

- Optimal cost function

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

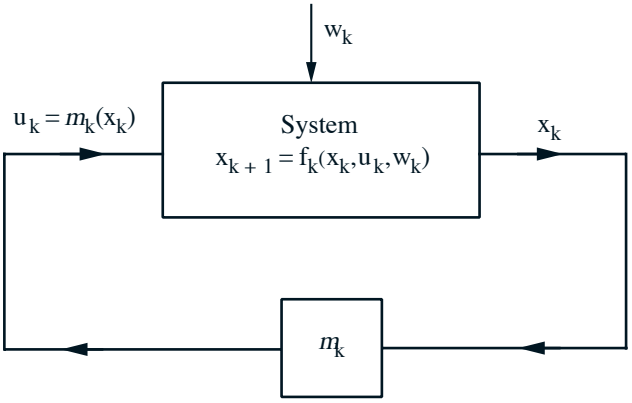
- Optimal policy π^* satisfies

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

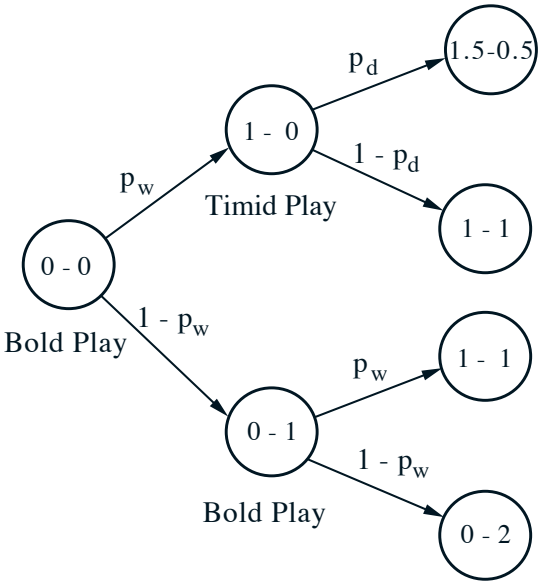
When produced by DP, π^* is independent of x_0.

SIGNIFICANCE OF FEEDBACK

- Open-loop versus closed-loop policies



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



A NOTE ON THESE SLIDES

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

6.231 DYNAMIC PROGRAMMING

LECTURE 2

LECTURE OUTLINE

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

BASIC PROBLEM

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

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

- Optimal cost function

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

- Optimal policy π^* is one that satisfies

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

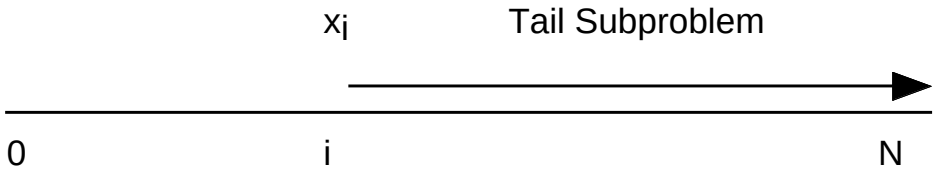
PRINCIPLE OF OPTIMALITY

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

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

and the "tail policy" $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$



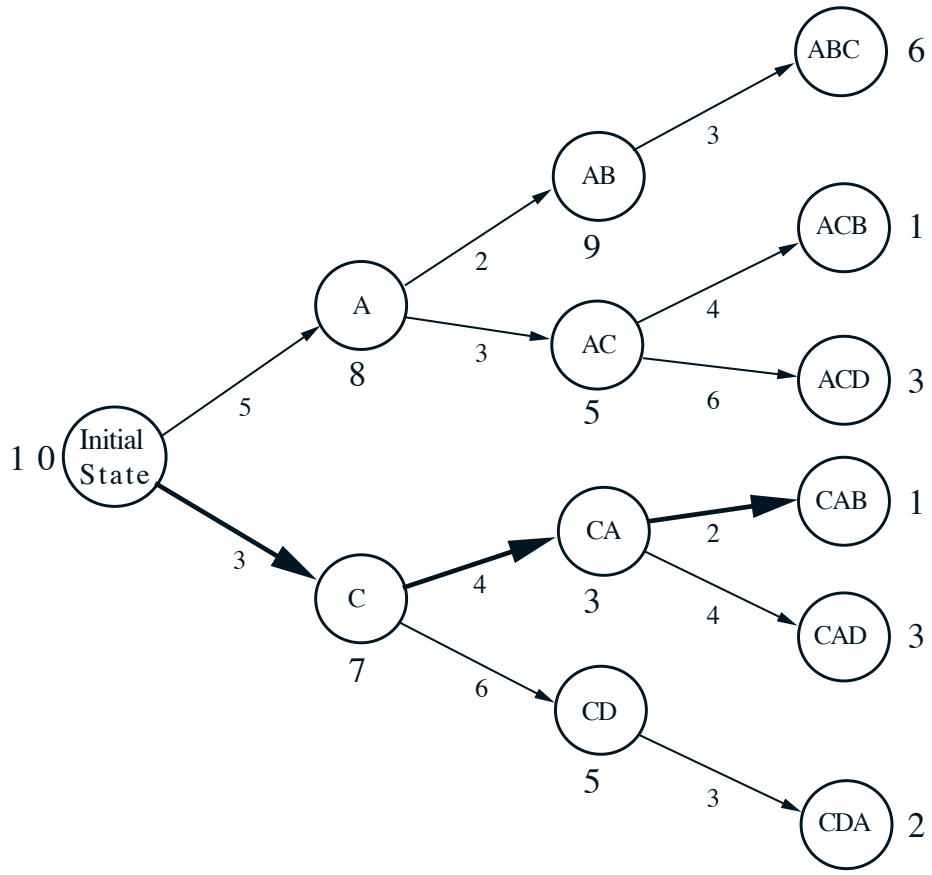
- *Principle of optimality*: The tail policy is optimal for the tail subproblem

- DP first solves ALL tail subroblems 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

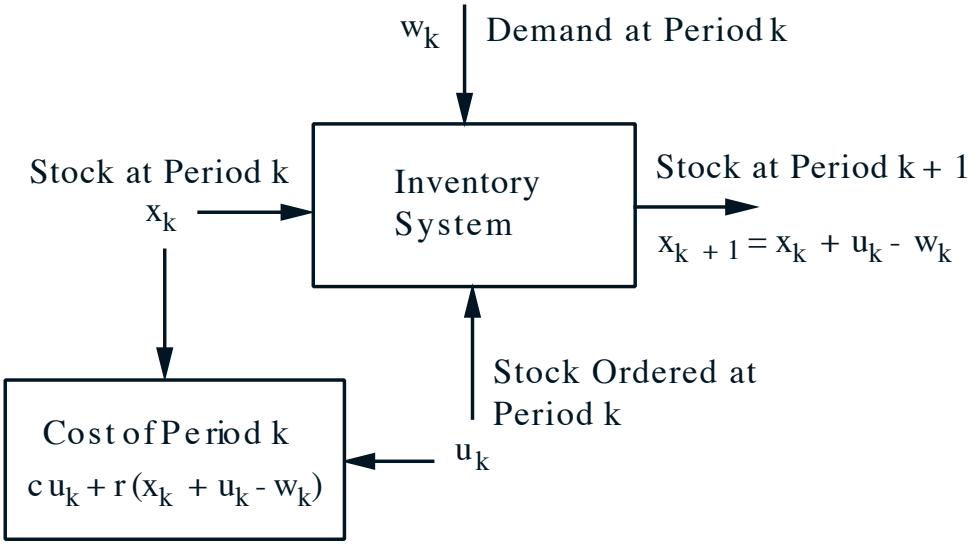
- Find optimal sequence of operations A, B, C, D (A must precede B and C must precede D)



- 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:

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

- Tail Subproblems of Length $N - k$:

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

DP ALGORITHM

- Start with

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

and go backwards using

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

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

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

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

- **Justification:** Proof by induction that $J_k(x_k)$ is equal to $J_k^*(x_k)$, defined as the optimal cost of the tail subproblem that starts at time k at state x_k.

- Note that ALL the tail subproblems are solved in addition to the original problem, and the intensive computational requirements.

PROOF OF THE INDUCTION STEP

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

$$J^*_k(x_k) = \min_{(\mu_k, \pi_{k+1})} \underset{w_k, \ldots, w_{N-1}}{E} \Bigg\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\big(x_i, \mu_i(x_i), w_i\big) \Bigg\}$$

$$= \min_{\mu_k} \underset{w_k}{E} \Bigg\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + \min_{\pi_{k+1}} \Bigg[\underset{w_{k+1}, \ldots, w_{N-1}}{E} \Bigg\{ g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\big(x_i, \mu_i(x_i), w_i\big) \Bigg\} \Bigg] \Bigg\}$$

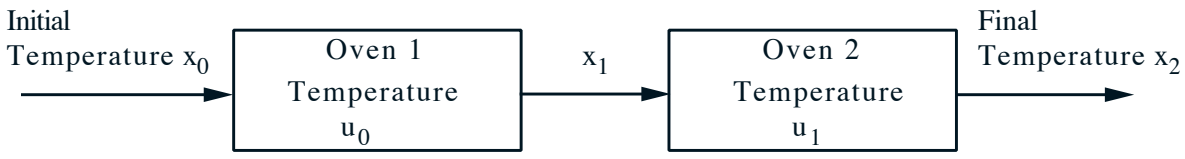
$$= \min_{\mu_k} \underset{w_k}{E} \Big\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + J^*_{k+1}\Big(f_k\big(x_k, \mu_k(x_k), w_k\big)\Big) \Big\}$$

$$= \min_{\mu_k} \underset{w_k}{E} \Big\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + J_{k+1}\Big(f_k\big(x_k, \mu_k(x_k), w_k\big)\Big) \Big\}$$

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

$$= J_k(x_k)$$

LINEAR-QUADRATIC ANALYTICAL EXAMPLE



- System

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

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:

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

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

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

STATE AUGMENTATION

- When assumptions of the basic problem are violated (e.g., disturbances are correlated, cost is nonadditive, etc) reformulate/augment 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:

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

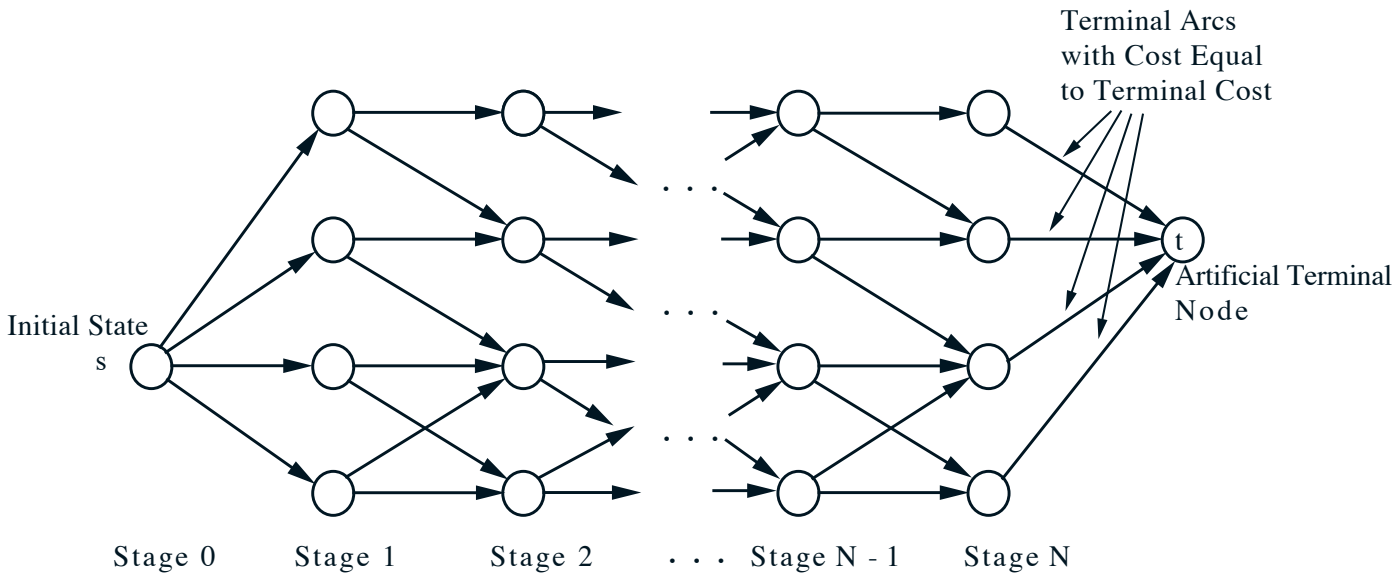
6.231 DYNAMIC PROGRAMMING

LECTURE 3

LECTURE OUTLINE

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

DETERMINISTIC FINITE-STATE PROBLEM



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

BACKWARD AND FORWARD DP ALGORITHMS

- DP algorithm:

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

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

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 $s \to t$ is also an optimal path $t \to s$ in a "reverse" shortest path problem where the direction of each arc is reversed and its length is left unchanged.

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

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

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

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

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

A NOTE ON FORWARD DP ALGORITHMS

- There is no forward DP algorithm for stochastic problems.

- Mathematically, for stochastic problems, we cannot restrict ourselves to open-loop sequences, so the shortest path viewpoint fails.

- Conceptually, in the presence of uncertainty, the concept of "optimal-cost-to-arrive" at a state x_k does not make sense. The reason is that it may be impossible to guarantee (with prob. 1) that any given state can be reached.

- By contrast, even in stochastic problems, the concept of "optimal cost-to-go" from any state x_k makes clear sense.

GENERIC SHORTEST PATH PROBLEMS

- $\{1, 2, \ldots, N, t\}$: 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.

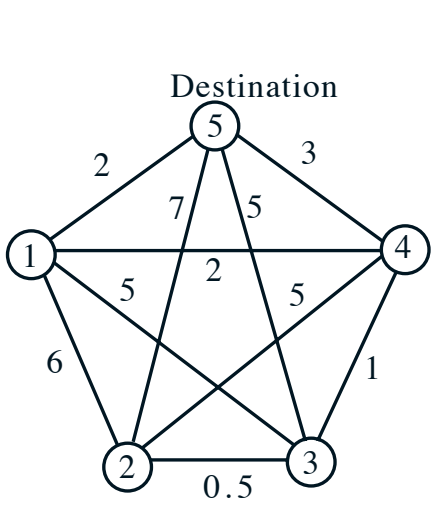
$J_0(i)$: Cost of the optimal path from i to t.

- DP algorithm:

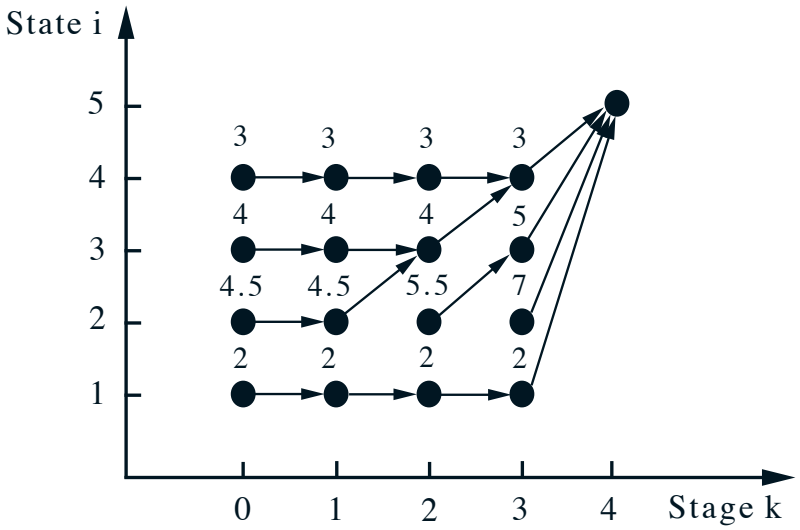
$$J_k(i) = \min_{j=1,\ldots,N} \left[a_{ij} + J_{k+1}(j)\right], \qquad k = 0, 1, \ldots, N-2,$$

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

EXAMPLE



(a)



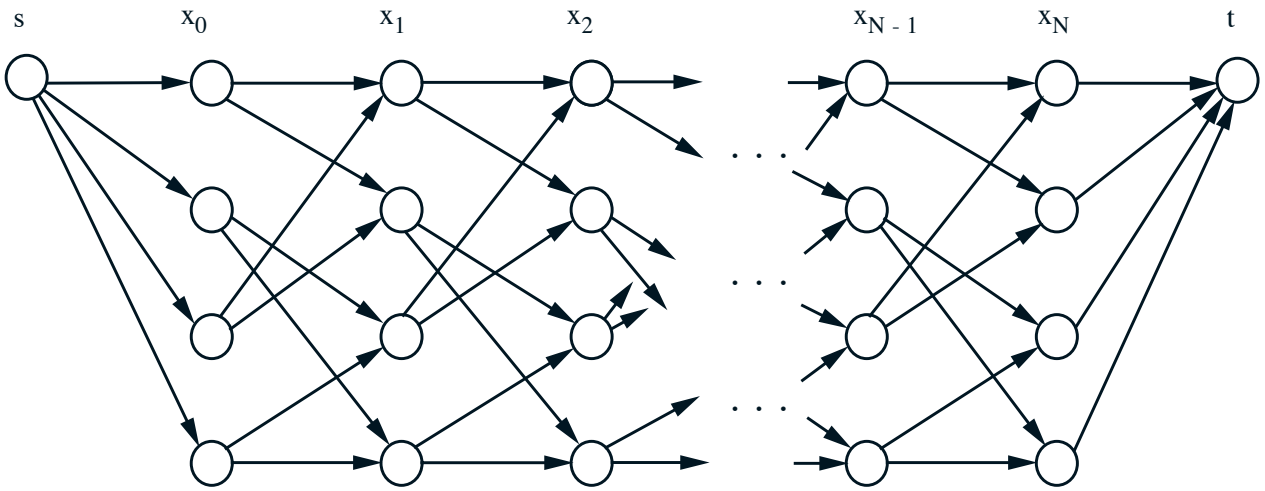
(b)

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

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

STATE ESTIMATION / HIDDEN MARKOV MODELS

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



VITERBI ALGORITHM

- We have

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

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 $p(X_N \mid Z_N)$ 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

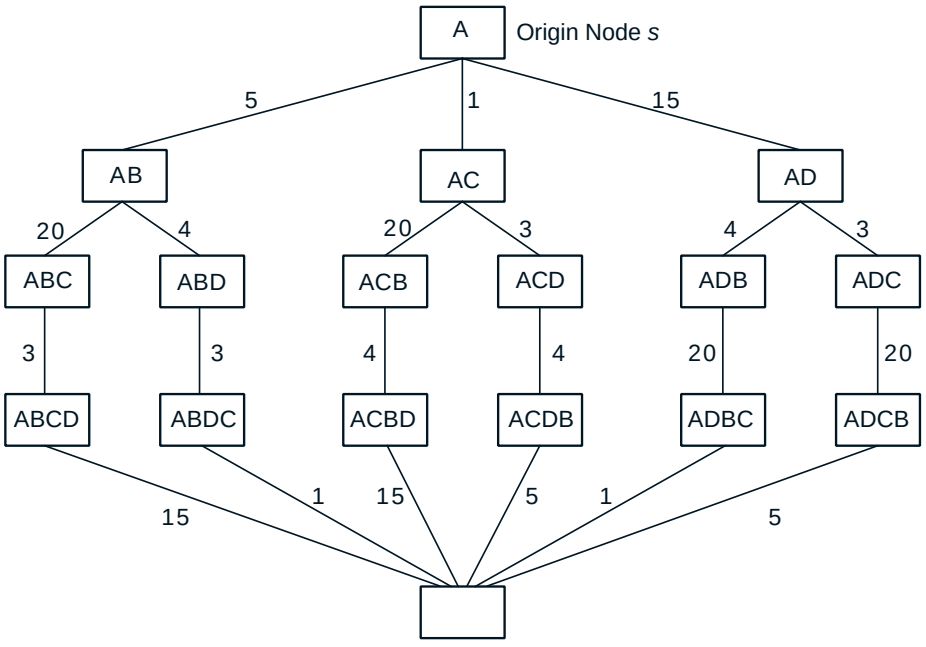
$$\text{minimize} - \ln(\pi_{x_0}) - \sum_{k=1}^{N} \ln\big(p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k)\big)$$

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

- This is a shortest path problem.

GENERAL SHORTEST PATH ALGORITHMS

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



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

LABEL CORRECTING METHODS

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

Label Correcting Algorithm

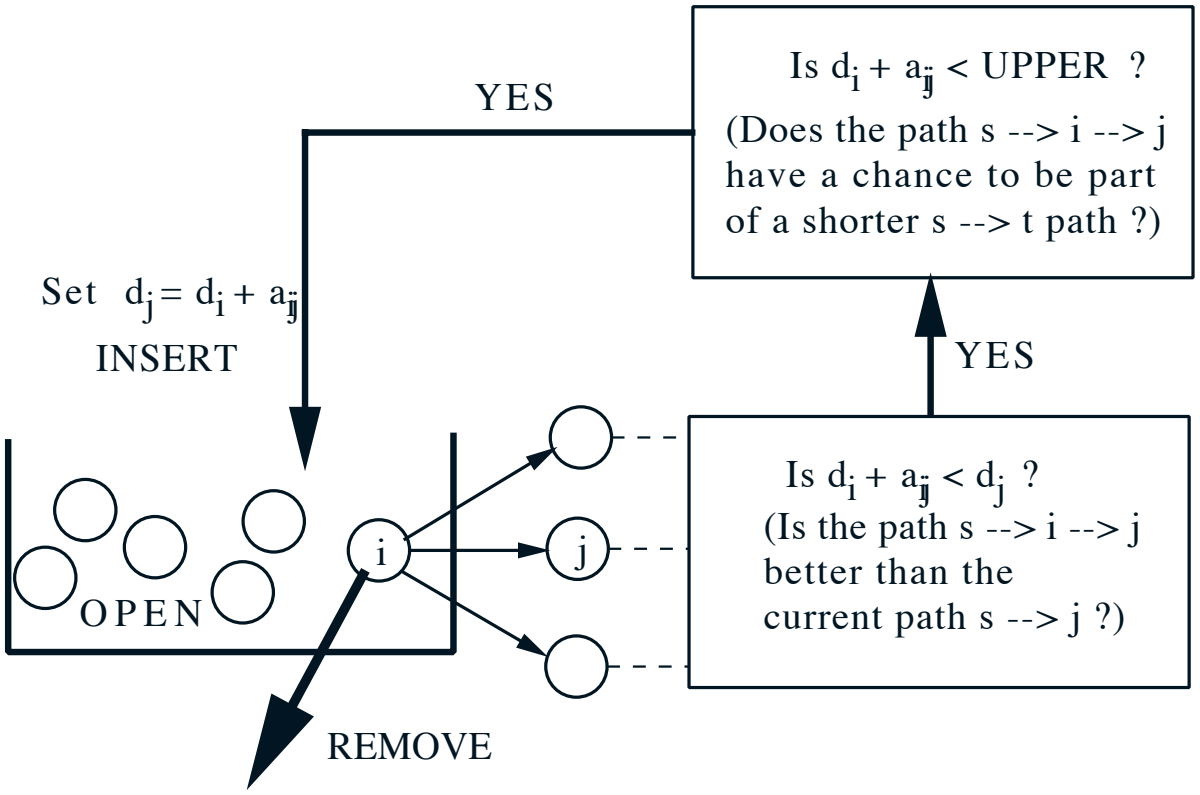
Step 1 (Node Removal): Remove a node i from OPEN and for each child j of i, do step 2.

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

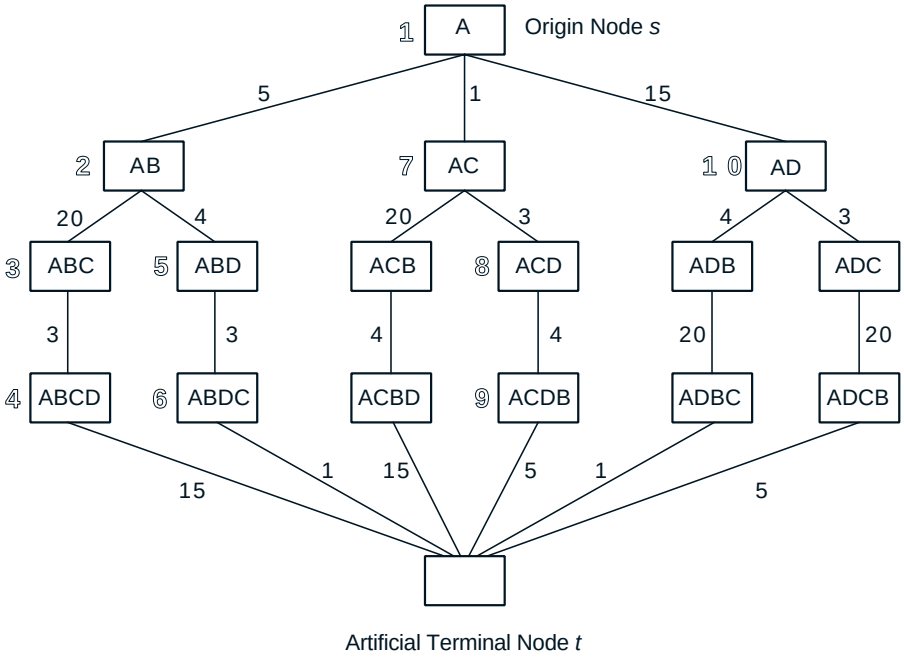
Step 3 (Termination Test): If OPEN is empty, terminate; else go to step 1.

VISUALIZATION/EXPLANATION

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



EXAMPLE

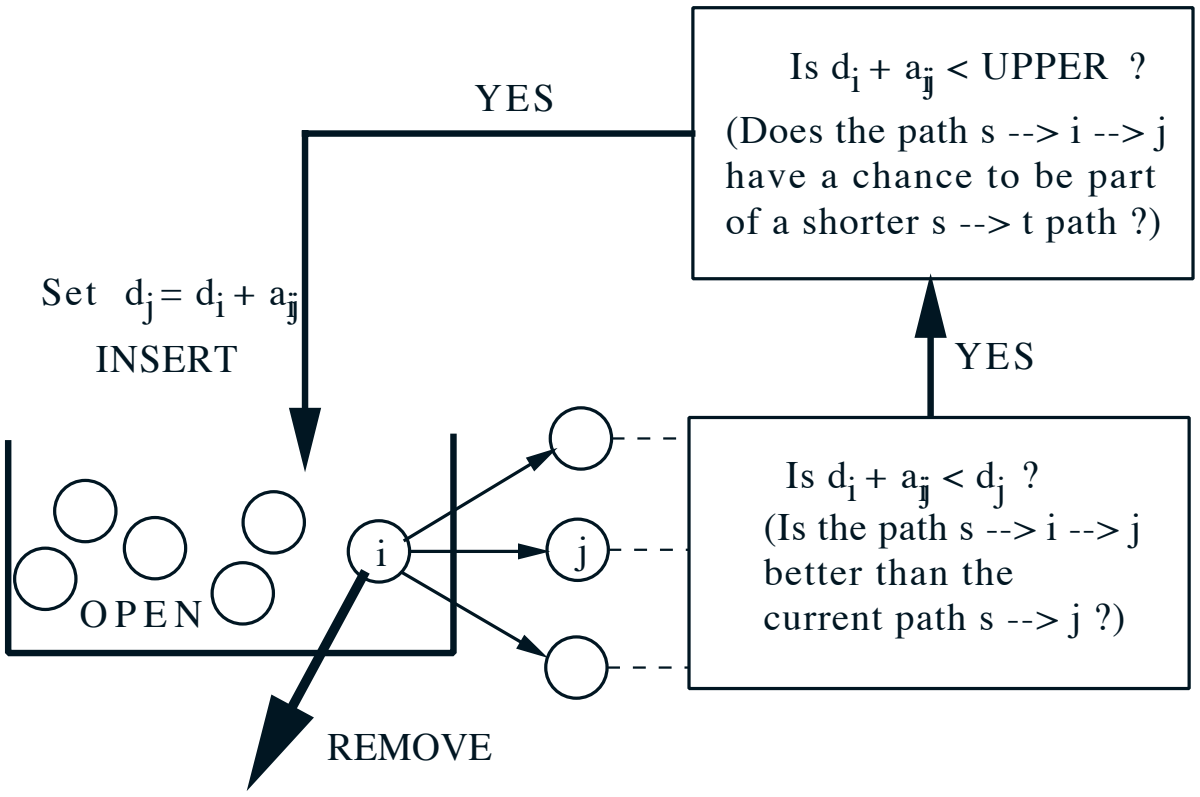


Iter. No.	Node Exiting OPEN	OPEN after Iteration	UPPER
0	-	1	∞
1	1	2, 7,10	∞
2	2	3, 5, 7, 10	∞
3	3	4, 5, 7, 10	∞
4	4	5, 7, 10	43
5	5	6, 7, 10	43
6	6	7, 10	13
7	7	8, 10	13
8	8	9, 10	13
9	9	10	13
10	10	Empty	13

- Note that some nodes never entered OPEN

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 $s \to i$ path.
- UPPER: Label d_t of the destination
- OPEN list: Contains "active" nodes (initially OPEN=$\{s\}$)



6.231 DYNAMIC PROGRAMMING

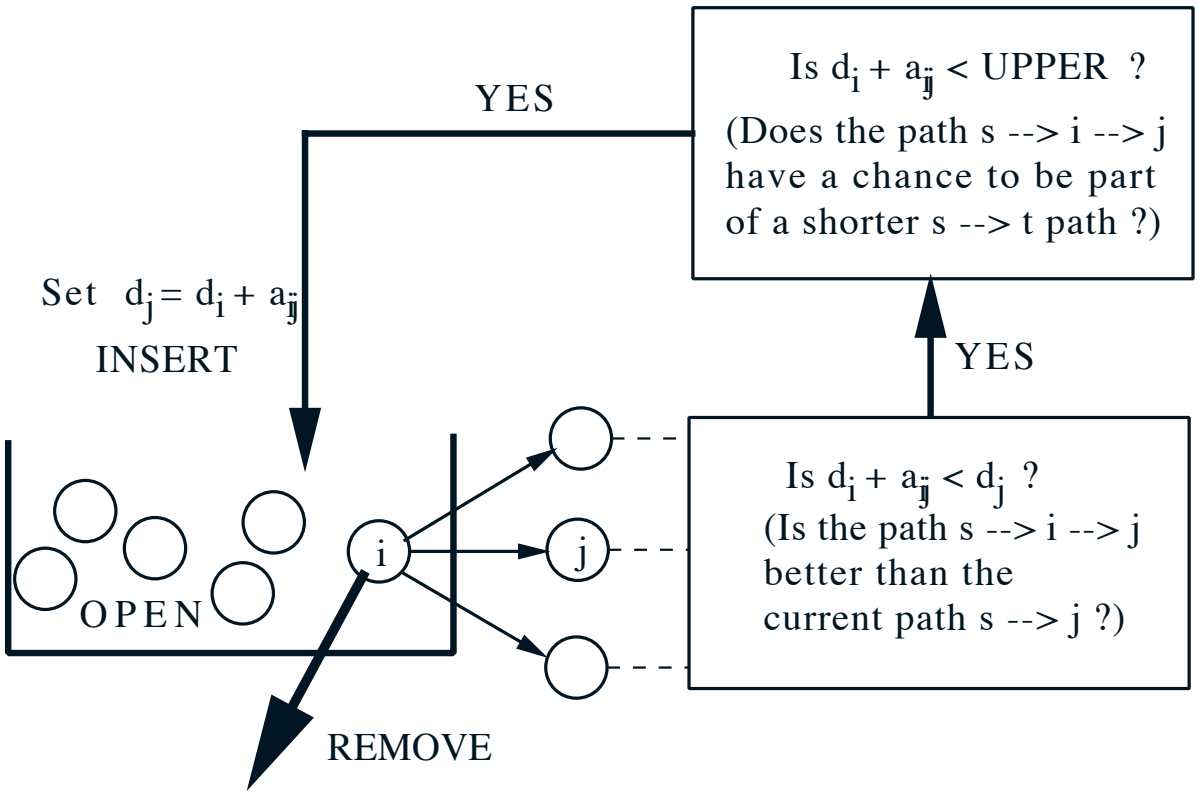
LECTURE 4

LECTURE OUTLINE

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

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 $s \to i$ path.
- UPPER: Label d_t of the destination
- OPEN list: Contains "active" nodes (initially OPEN=$\{s\}$)



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

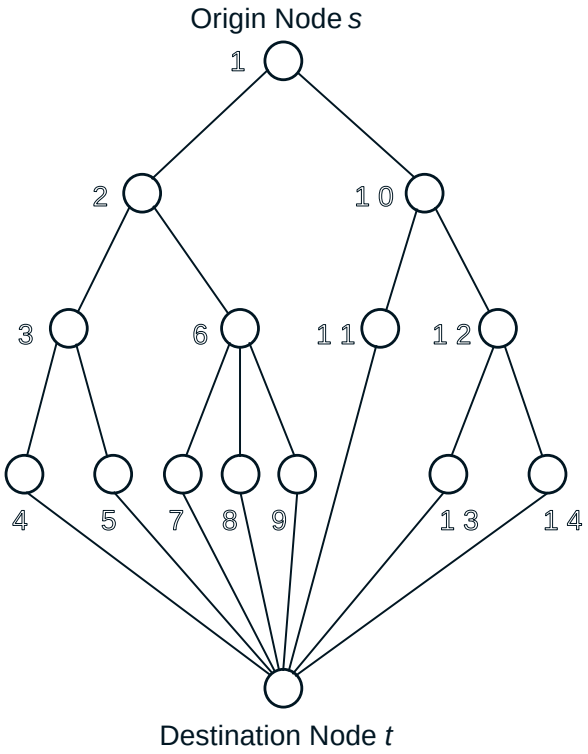
(3) Let $(s, j_1, j_2, \ldots, j_k, t)$ be a shortest path and let d^* be the shortest distance. If UPPER $> d^*$ at termination, UPPER will also be larger than the length of all the paths $(s, j_1, \ldots, j_m)$, $m = 1, \ldots, k$, throughout the algorithm. Hence, node j_k will never enter the OPEN 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.

MAKING THE METHOD EFFICIENT

- Reduce the value of UPPER as quickly as possible
  - Try to discover "good" $s \to t$ 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.
  - Heuristic rationale: if d_i is small, then d_j when set to $d_i + a_{ij}$ will be accordingly small, so reentrance of j in the OPEN list is less likely.
- 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
- Good practical strategies try to strike a compromise between low overhead and small label node selection.

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 (Djikstra):** 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 $\geq$ 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

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

The label correcting method with lower bounds as above is often referred to as the A^* **method.**

- If an **upper bound** m_j of the true shortest distance from j to t is known, then if $d_j + m_j <$ UPPER, reduce UPPER to $d_j + m_j$.

- **Important use:** Branch-and-bound algorithm for discrete optimization can be viewed as an implementation of this last variant.

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

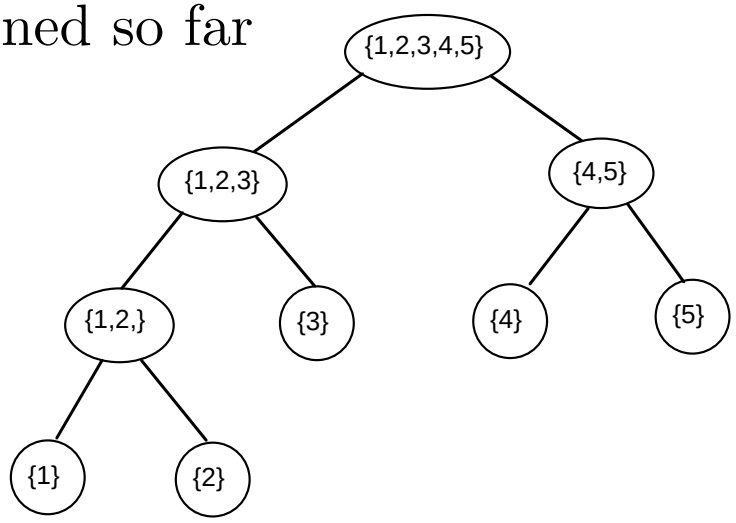
$$\underline{f}_1 \leq \min_{x \in Y_1} f(x), \qquad \overline{f}_2 \geq \min_{x \in Y_2} f(x).$$

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

- The B+B algorithm can be viewed as a label correcting algorithm, where lower bounds define the arc costs, and upper bounds are used to strengthen the test for admission to OPEN.

SHORTEST PATH IMPLEMENTATION

- Acyclic graph/partition of X into subsets (typically a tree). The leafs consist of single solutions.
- Upper/Lower bounds $\underline{f}_Y$ and $\overline{f}_Y$ for the minimum cost over each subset Y can be calculated.
- The lower bound of a leaf $\{x\}$ is $f(x)$
- Each arc (Y, Z) has length $\underline{f}_Z - \underline{f}_Y$
- Shortest distance from X to $Y = \underline{f}_Y - \underline{f}_X$
- Distance from origin X to a leaf $\{x\}$ is $f(x) - \underline{f}_X$
- Distance from origin X to a leaf $\{x\}$ is $f(x) - \underline{f}_X$
- Shortest path from X to the set of leafs gives the optimal cost and optimal solution
- UPPER is the smallest $f(x)$ out of leaf nodes $\{x\}$ examined so far



BRANCH-AND-BOUND ALGORITHM

Step 1: Remove a node Y from OPEN. For each child Y_j of Y, do the following: If $\underline{f}_{Y_j} <$ UPPER, then place Y_j in OPEN. If in addition $\overline{f}_{Y_j} <$ UPPER, then set UPPER $= \overline{f}_{Y_j}$, 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 (generate it as you 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.

6.231 DYNAMIC PROGRAMMING

LECTURE 5

LECTURE OUTLINE

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

LINEAR-QUADRATIC PROBLEMS

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

$$\underset{\substack{w_k \\ k=0,1,\ldots,N-1}}{E} \left\{ x_N' Q_N x_N + \sum_{k=0}^{N-1} (x_k' Q_k x_k + u_k' R_k u_k) \right\}$$

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

- w_k are independent and zero mean
- DP algorithm:

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

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

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

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

  - Similar treatment of a number of variants

DERIVATION

- By induction verify that

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

where L_k are matrices given by

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

and where K_k are symmetric positive semidefinite matrices given by

$$K_N = Q_N,$$

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

- 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: controlability of (A, B) and observability of (A, C) where $Q = C'C$

• The Riccati equation converges $\lim_{k\to-\infty} K_k = K$, where K is pos. definite, and is the unique (within the class of pos. semidefinite matrices) solution of the *algebraic Riccati equation*

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

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

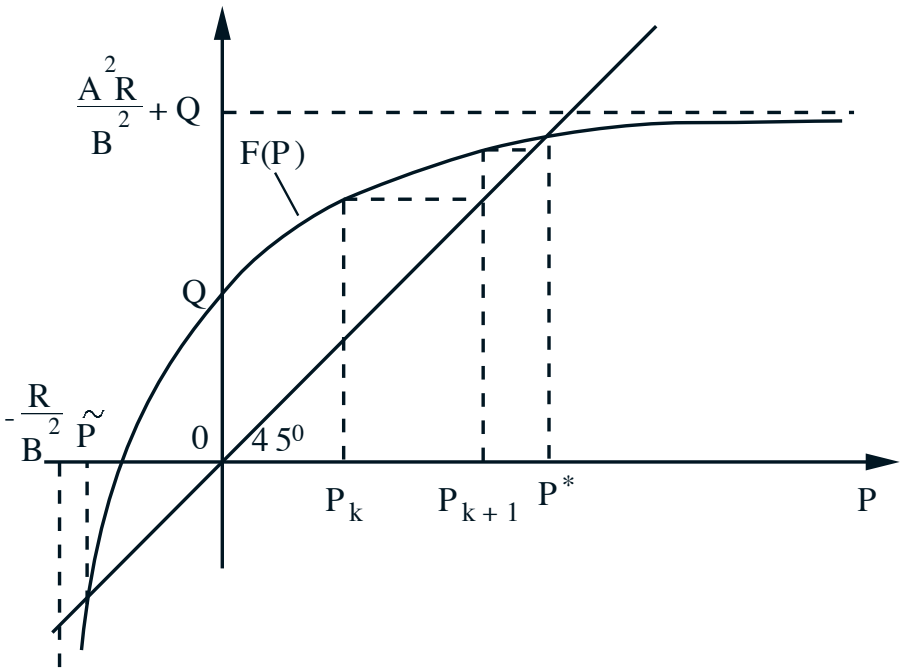
$$L = -(B'KB + R)^{-1}B'KA,$$

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

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

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

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

RANDOM SYSTEM MATRICES

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

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

$$J_k(x_k) = \min_{u_k} \underset{w_k, A_k, B_k}{E} \Big\{ x_k' Q_k x_k + u_k' R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k) \Big\}$$

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

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

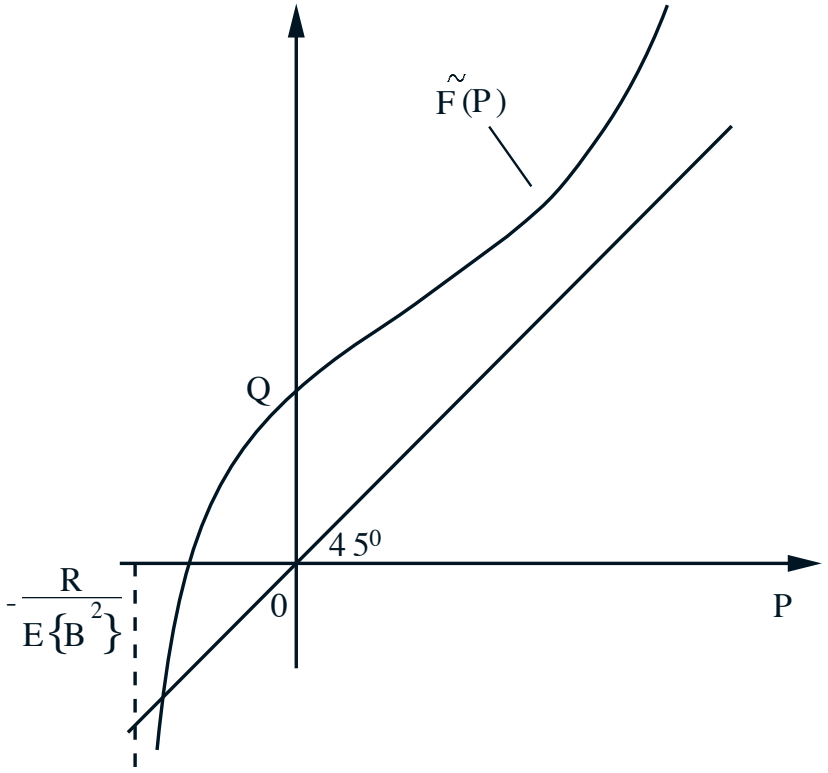
and where the matrices K_k are given by

$$K_N = Q_N,$$

$$K_k = E\{A_k' K_{k+1} A_k\} - E\{A_k' K_{k+1} B_k\} \big(R_k + E\{B_k' K_{k+1} B_k\}\big)^{-1} E\{B_k' K_{k+1} A_k\} + Q_k$$

PROPERTIES

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



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

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

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

INVENTORY CONTROL

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

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

- Minimize

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

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

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

- DP algorithm:

$$J_N(x_N) = 0,$$

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

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

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

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

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

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

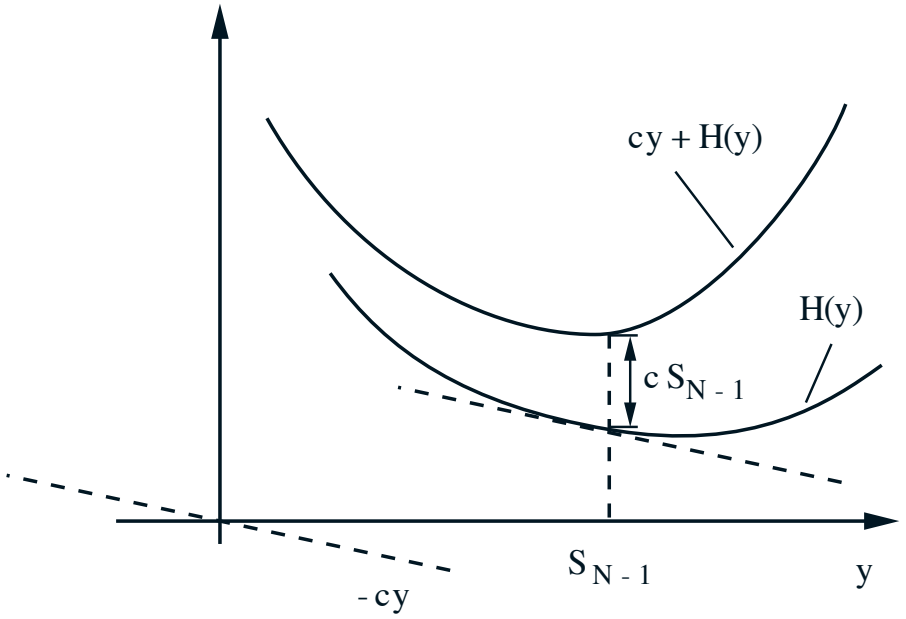
where S_k minimizes $G_k(y)$.

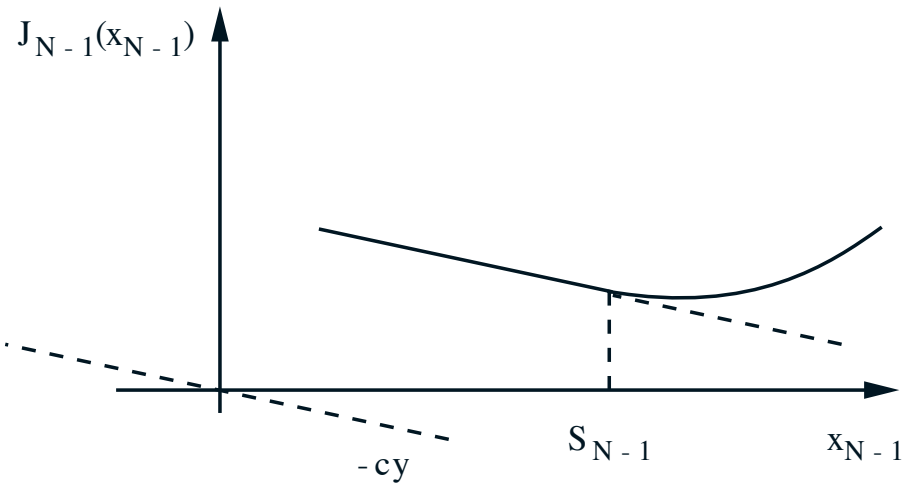
- This is shown, assuming that $c < p$, by showing that J_k is convex for all k, and

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

JUSTIFICATION

- Graphical inductive proof that J_k is convex.





6.231 DYNAMIC PROGRAMMING

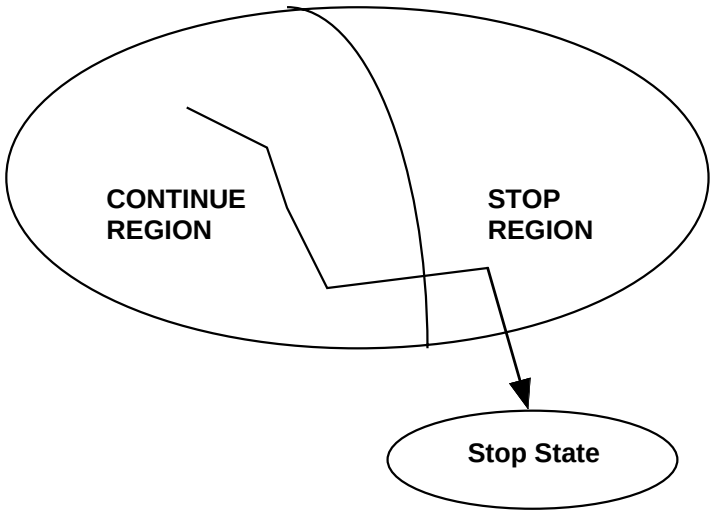
LECTURE 6

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 $k = 0, 1, \ldots, N-1$ 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):

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

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

- Optimal policy;

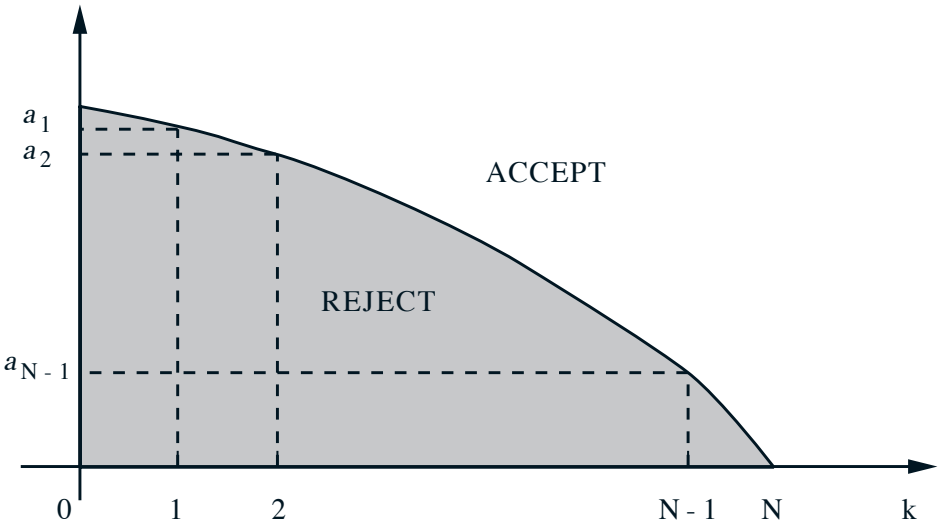
$$\text{accept the offer } x_k \qquad \text{if } x_k > \alpha_k,$$

$$\text{reject the offer } x_k \qquad \text{if } x_k < \alpha_k,$$

where

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

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

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

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 $\alpha_k \to \overline{a}$ as $k \to -\infty$. 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

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

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

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

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

- 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

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

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 kth and $(k+1)$st questions in an optimally ordered list

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

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

Consider the list with i and j interchanged

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

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

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

MINIMAX CONTROL

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

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

- The DP algorithm takes the form

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

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

(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

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

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

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

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

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

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

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 as a continuous-time limit of the DP algorithm
- The Hamilton-Jacobi-Bellman equation as a sufficient condition
- Examples

PROBLEM FORMULATION

- We have a continuous-time dynamic system

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

where
  - $x(t) \in \Re^n$ is the state vector at time t
  - $u(t) \in U \subset \Re^m$ is the control vector at time t, U is the control constraint set
  - T is the terminal time.

- Any admissible control trajectory $\big\{u(t) \mid t \in [0,T]\big\}$ (piecewise continuous function $\big\{u(t) \mid t \in [0,T]\big\}$ with $u(t) \in U$ for all $t \in [0,T]$), uniquely determines $\big\{x(t) \mid t \in [0,T]\big\}$.

- Find an admissible control trajectory $\big\{u(t) \mid t \in [0,T]\big\}$ and corresponding state trajectory $\big\{x(t) \mid t \in [0,T]\big\}$, that minimizes a cost function of the form

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

- 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.
- $x(t) = \big(x_1(t), x_2(t)\big)$: position and velocity of the mass at time t
- Problem: From a given $\big(x_1(0), x_2(0)\big)$, bring the mass "near" a given final position-velocity pair $(\overline{x}_1, \overline{x}_2)$ at time T in the sense:

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

subject to the control constraint

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

- The problem fits the framework with

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

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

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

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

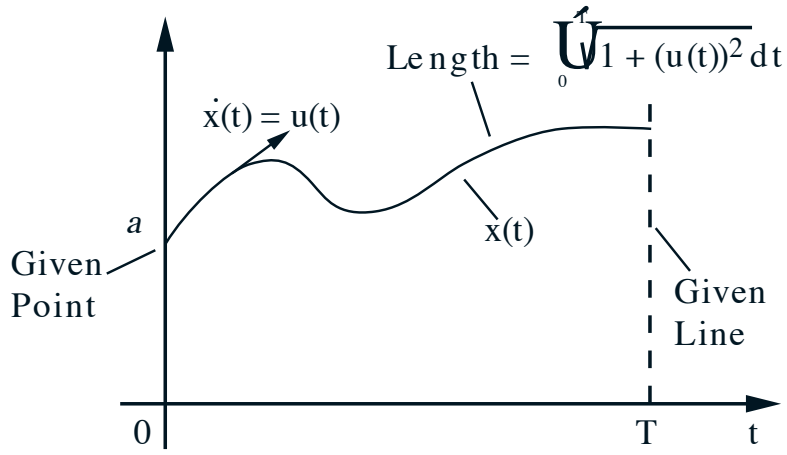
$$\int_0^T \big(1 - u(t)\big) x(t) dt$$

subject to

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

• 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

$$\text{minimize} \quad \int_0^T \sqrt{1 + \big(\dot{x}(t)\big)^2}\, dt$$
$$\text{subject to} \quad x(0) = \alpha.$$

- Reformulation as an optimal control problem:

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

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

HAMILTON-JACOBI-BELLMAN EQUATION I

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

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

- 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

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

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

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

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

HAMILTON-JACOBI-BELLMAN EQUATION II

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

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

we obtain for all t,x,

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

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/SUFFICIENCY THEOREM

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

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

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

- Suppose also that $\mu^*(t,x)$ attains the minimum above for all t and x.

- Let $\big\{ x^*(t) \mid t \in [0,T] \big\}$ and $u^*(t) = \mu^*\big(t, x^*(t)\big)$, $t \in [0,T]$, be the corresponding state and control trajectories.

- Then

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

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

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

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

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

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

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

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

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

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

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

EXAMPLE OF THE HJB EQUATION

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

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

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

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

- 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

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

LINEAR QUADRATIC PROBLEM

Consider the n-dimensional linear system

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

and the quadratic cost

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

The HJB equation is

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

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

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

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

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

- Cost function

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

- $J^*(t,x)$: optimal cost-to-go from x at time t
- HJB equation: For all (t,x)

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

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 $\mu^*(t,x)$ such that

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

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 $t \in [0,T]$

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

- 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

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

- 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. $(*)$ 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 $t \in \Re$, $x \in \Re^n$, and $u \in \Re^m$, and let U be a convex subset of $\Re^m$. Assume that $\mu^*(t, x)$ is a continuously differentiable function such that

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

Then

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

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

DIFFERENTIATING THE HJB EQUATION I

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

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

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),

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

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

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

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

DIFFERENTIATING THE HJB EQUATION II

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

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

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

are equal to the total derivatives

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

and we have

$$0 = \nabla_x g\big(x, u^*(t)\big) + \frac{d}{dt}\big(\nabla_x J^*\big(t, x^*(t)\big)\big) + \nabla_x f\big(x, u^*(t)\big) \nabla_x J^*\big(t, x^*(t)\big)$$

$$0 = \frac{d}{dt}\big(\nabla_t J^*\big(t, x^*(t)\big)\big).$$

CONCLUSION FROM DIFFERENTIATING THE HJB

- Define

$$p(t) = \nabla_x J^*\big(t, x^*(t)\big)$$

and

$$p_0(t) = \nabla_t J^*\big(t, x^*(t)\big)$$

- We have the *adjoint equation*

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

and

$$\dot{p}_0(t) = 0$$

or equivalently,

$$p_0(t) = \text{constant}, \qquad \text{for all } t \in [0, T].$$

- Note also that, by definition $J^*\big(T, x^*(T)\big) = h\big(x^*(T)\big)$, so we have the following boundary condition at the terminal time:

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

NOTATIONAL SIMPLIFICATION

- Define the *Hamiltonian* function

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

- The adjoint equation becomes

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

- The HJB equation becomes

$$\begin{aligned} 0 &= \min_{u \in U} \big[H\big(x^*(t), u, p(t)\big) \big] + p_0(t) \\ &= H\big(x^*(t), u^*(t), p(t)\big) + p_0(t) \end{aligned}$$

so since $p_0(t) =$ constant, there is a constant C such that

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

PONTRYAGIN MINIMUM PRINCIPLE

- The preceding (highly informal) derivation is summarized as follows:

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

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

with the boundary condition

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

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

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

Furthermore, there is a constant C such that

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

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

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

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

with split boundary conditions:

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

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

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

- This 2-point boundary value problem can be addressed with a variety of numerical methods.

ANALYTICAL EXAMPLE I

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

subject to

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

- 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

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

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.

ANALYTICAL EXAMPLE II

- Optimal production problem

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

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

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

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

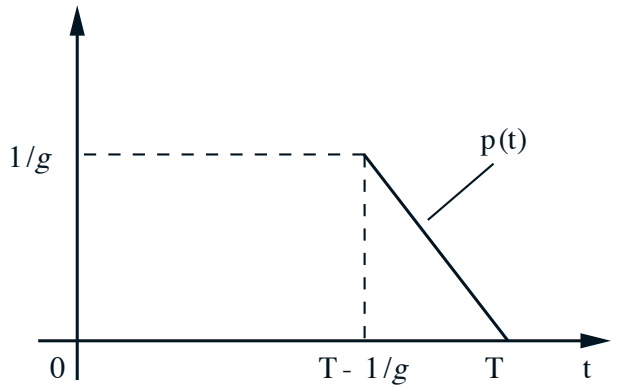
$$\dot{p}(t) = -\gamma u^*(t)p(t) - 1 + u^*(t), \quad p(T) = 0.$$

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

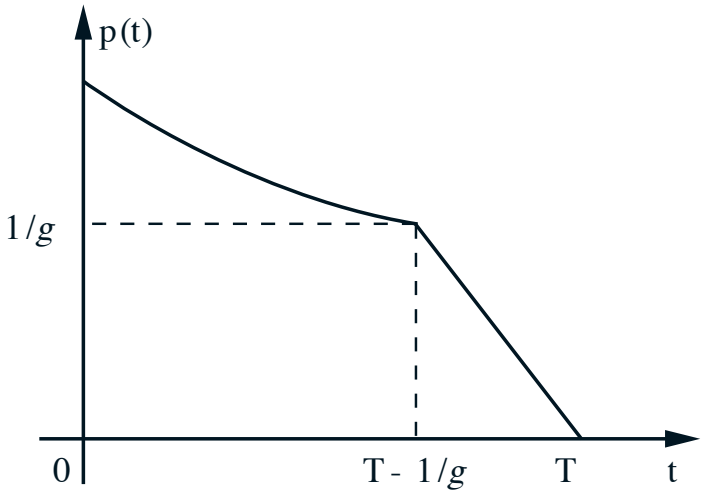
$$u^*(t) = \begin{cases} 0 & \text{if } p(t) < \frac{1}{\gamma}, \\ 1 & \text{if } p(t) \geq \frac{1}{\gamma}. \end{cases}$$

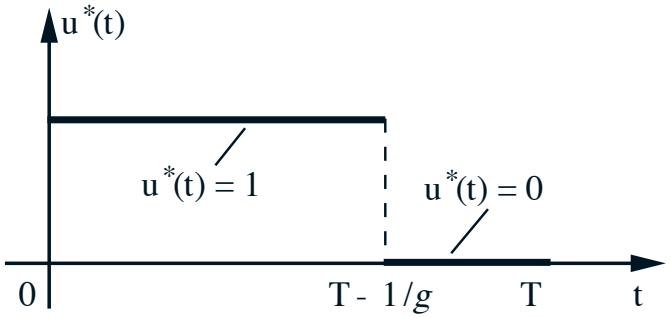
Since $p(T) = 0$, for t close to T, $p(t) < 1/\gamma$ and $u^*(t) = 0$. Therefore, for t near T the adjoint equation has the form $\dot{p}(t) = -1$.

ANALYTICAL 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

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

- Cost function

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

- $J^*(t,x)$: optimal cost-to-go from x at time t

- HJB equation/Verification theorem: For all (t,x)

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

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

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

- 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

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

with the boundary condition

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

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

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

Furthermore, there is a constant C such that

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

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

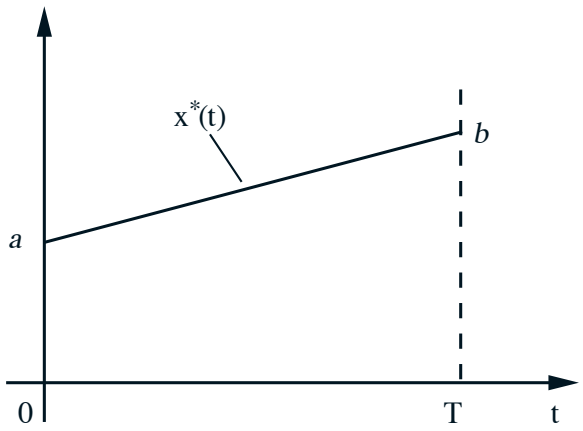
- Generalization: Some components of the terminal state are fixed.

EXAMPLE WITH FIXED TERMINAL STATE

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

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

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



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

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

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

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

So optimal $\big\{x^*(t) \mid t \in [0, T]\big\}$ 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 $\big\{\big(x^*(t), u^*(t)\big) \mid t \in [0, T]\big\}$ 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

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

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

• In addition: $H(x^*(t), u^*(t), p(t)) = 0$ for all t [instead of $H(x^*(t), u^*(t), p(t)) \equiv$ constant].

• Justification: We have

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

Along the optimal, the HJB equation is

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

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

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

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

so the system equation is

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

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

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

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.,

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

Therefore

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

• The adjoint equation is

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

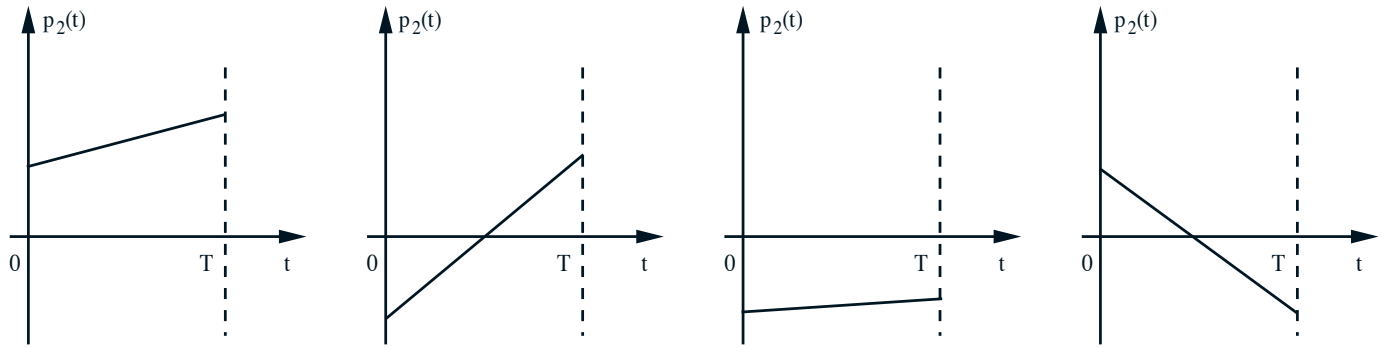
so

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

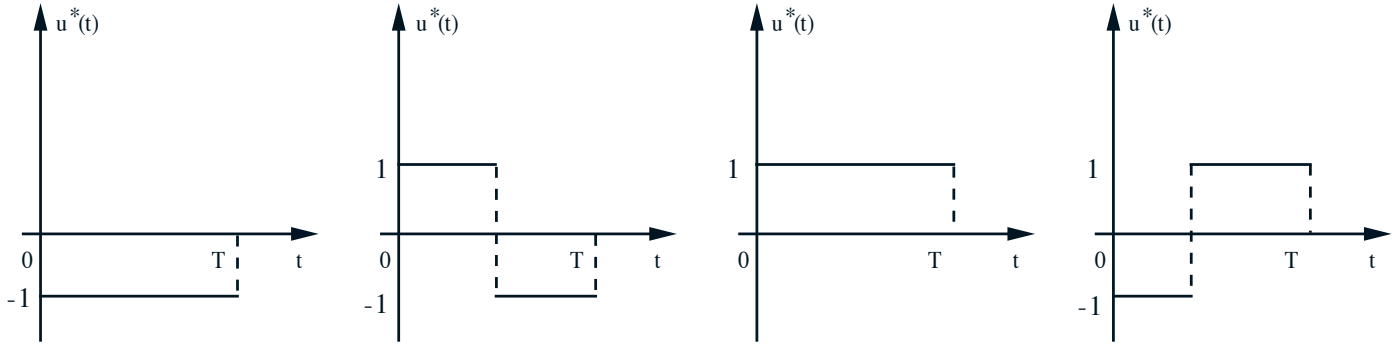
where c_1 and c_2 are constants.

• 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

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

Eliminating the time t, we see that for all t

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

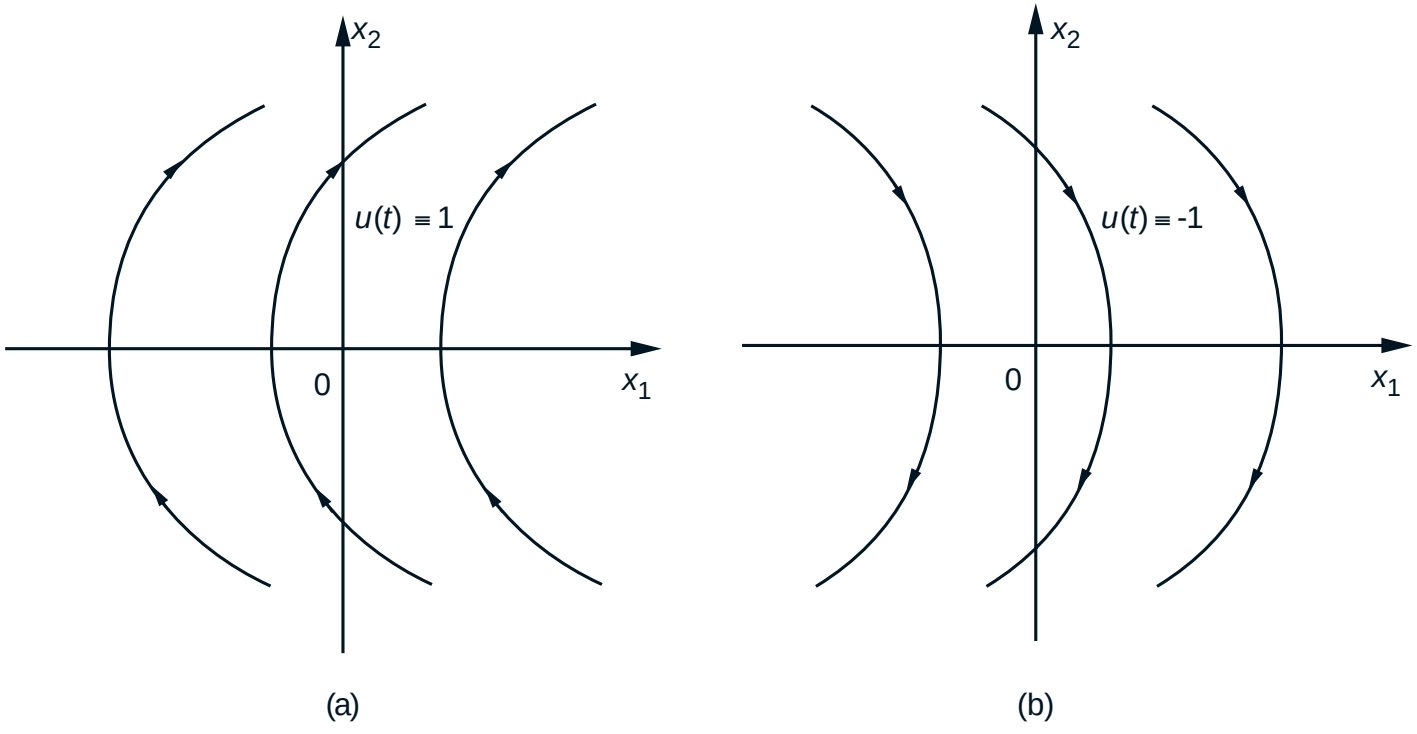
MINIMUM-TIME EXAMPLE IV

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

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

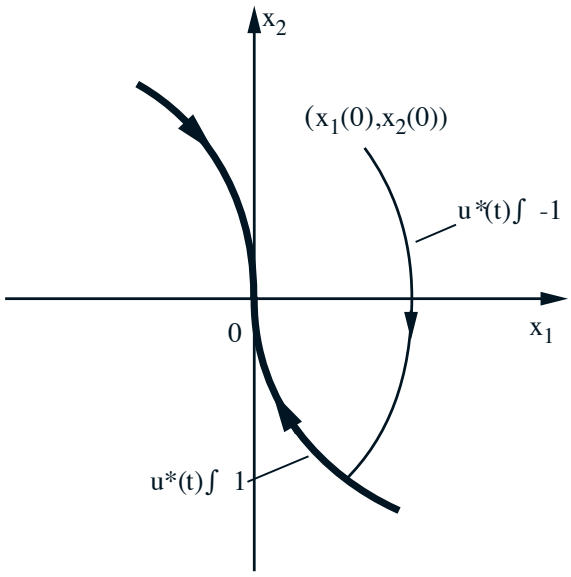
- For intervals where $u(t) \equiv -1$, the system moves along the curves

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



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.



(a) If the initial state lies *above* the switching curve, use $u^*(t) \equiv -1$ until the state hits the switching curve; then use $u^*(t) \equiv 1$.

(b) If the initial state lies *below* the switching curve, use $u^*(t) \equiv 1$ until the state hits the switching curve; then use $u^*(t) \equiv -1$.

(c) 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 $u_k \in U_k \subset \Re^m$, with U_k: convex, and

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

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

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

where $p_1, \ldots, p_N$ 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

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

DERIVATION

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

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

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

• Iintroduce the discrete-time adjoint equation

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

with terminal condition $p_N = \nabla g_N$.

• Verify that, for all k,

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

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

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

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

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

- 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 $w_0, \ldots, w_{k-1}, v_0, \ldots, v_{k-1}$.
- 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:

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

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

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

- We want to find a policy π that minimizes

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

subject to the equations

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

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

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 kth 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 (as well as $u_0, u_1, \ldots, u_{k-1}$). Mathematically, z_{k+1}, the observation at the end of the kth 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

$$J_k(x_k) = g_k(x_k) + \min_{0 \le u_k \le 1} \underset{a_k}{E} \Big\{ p(x_k, u_k) J_{k+1}(x_k + a_k - 1) + \big(1 - p(x_k, u_k)\big) J_{k+1}(x_k + a_k) \Big\},$$

$p(x_k, u_k)$ 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 $x_k \ge 1$.

- Imperfect state info problem is much harder.

REFORMULATION AS A PERFECT INFO PROBLEM

- We have

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

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

- We have

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

since $z_0, z_1, \ldots, z_k$ 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" $z_k, \ldots, z_0$.

- Write

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

so the cost per stage of the new system is

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

DP ALGORITHM

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

$$J_k(I_k) = \min_{u_k \in U_k} \Bigg[\underset{x_k,\, w_k,\, z_{k+1}}{E} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \Big\} \Bigg]$$

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

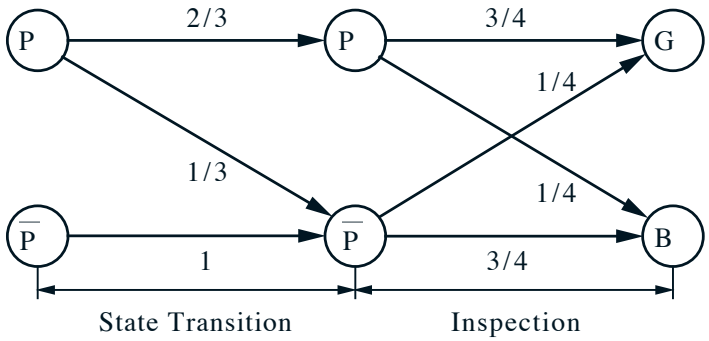
$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \Bigg[\underset{x_{N-1},\, w_{N-1}}{E} \Big\{ g_N\Big(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\Big) + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) \mid I_{N-1}, u_{N-1} \Big\} \Bigg],$$

- The optimal cost J^* is given by

$$J^* = \underset{z_0}{E}\Big\{ J_0(z_0) \Big\}.$$

MACHINE REPAIR EXAMPLE I

- A machine can be in one of two states denoted P (good state) and $\overline{P}$ (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 $\overline{P}$ bring it back to the good state P.

- Cost per stage:

$$g(P,C)=0,\quad g(P,S)=1,\quad g(\overline{P},C)=2,\quad g(\overline{P},S)=1.$$

MACHINE REPAIR EXAMPLE II

- The information vector at times 0 and 1 is

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

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

$$\underset{\substack{x_0,\, w_0,\, w_1 \\ v_0,\, v_1}}{E} \Big\{ g\big(x_0, \mu_0(z_0)\big) + g\big(x_1, \mu_1(z_0, z_1, \mu_0(z_0))\big) \Big\}.$$

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

$$\begin{aligned} J_k(I_k) = \min \Big[& P(x_k = P \mid I_k) g(P, C) \\ & + P(x_k = \overline{P} \mid I_k) g(\overline{P}, C) \\ & + \underset{z_{k+1}}{E} \Big\{ J_{k+1}(I_k, C, z_{k+1}) \mid I_k, C \Big\}, \\ & P(x_k = P \mid I_k) g(P, S) \\ & + P(x_k = \overline{P} \mid I_k) g(\overline{P}, S) \\ & + \underset{z_{k+1}}{E} \Big\{ J_{k+1}(I_k, S, z_{k+1}) \mid I_k, S \Big\} \Big] \end{aligned}$$

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

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

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

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

$$P(x_1 = \overline{P} \mid G, G, S) = \frac{P(x_1 = \overline{P}, G, G \mid S)}{P(G, G \mid S)}$$

$$= \frac{\frac{1}{3} \cdot \frac{1}{4} \cdot \left(\frac{2}{3} \cdot \frac{3}{4} + \frac{1}{3} \cdot \frac{1}{4}\right)}{\left(\frac{2}{3} \cdot \frac{3}{4} + \frac{1}{3} \cdot \frac{1}{4}\right)^2} = \frac{1}{7}.$$

Hence

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

MACHINE REPAIR EXAMPLE IV

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

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

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

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

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

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

• 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

$$\begin{aligned}\text{cost of } C &= 2P(x_0 = \overline{P} \mid I_0) + \mathop{E}_{z_1}\Big\{ J_1(I_0, z_1, C) \mid I_0, C \Big\} \\ &= 2P(x_0 = \overline{P} \mid I_0) + P(z_1 = G \mid I_0, C) J_1(I_0, G, C) \\ &\quad + P(z_1 = B \mid I_0) J_1(I_0, B, C),\end{aligned}$$

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

using the values of J_1 from the previous stage.

- We have

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

- 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

EVIEW: PROBLEM WITH IMPERFECT STATE INF

- Instead of knowing x_k, we receive observations

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

- I_k: information vector available at time k:

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

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

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

subject to the equations

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

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

DP ALGORITHM

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

$$J_k(I_k) = \min_{u_k \in U_k} \Big[\mathop{E}_{x_k,\, w_k,\, z_{k+1}} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \Big\} \Big]$$

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

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \Big[\mathop{E}_{x_{N-1},\, w_{N-1}} \Big\{ g_N\big(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\big) + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) \mid I_{N-1}, u_{N-1} \Big\} \Big],$$

- The optimal cost J^* is given by

$$J^* = \mathop{E}_{z_0} \big\{ J_0(z_0) \big\}.$$

LINEAR-QUADRATIC PROBLEMS

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

$$\underset{\substack{w_k \\ k=0,1,\ldots,N-1}}{E} \left\{ x_N' Q_N x_N + \sum_{k=0}^{N-1} (x_k' Q_k x_k + u_k' R_k u_k) \right\}$$

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

- Observations

$$z_k = C_k x_k + v_k, \qquad k = 0, 1, \ldots, N-1.$$

- $w_0, \ldots, w_{N-1}$, $v_0, \ldots, v_{N-1}$ indep. zero mean
- Key fact to show:
  - Optimal policy $\{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is of the form:

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

    L_k: same as for the perfect state info case
  - Estimation problem and control problem can be solved separately

DP ALGORITHM I

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

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1}} \Big[E_{x_{N-1},w_{N-1}} \big\{ x'_{N-1} Q x_{N-1} + u'_{N-1} R u_{N-1} + (Ax_{N-1} + Bu_{N-1} + w_{N-1})' \cdot Q(Ax_{N-1} + Bu_{N-1} + w_{N-1}) \mid I_{N-1}, u_{N-1} \big\} \Big]$$

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

$$\min_{u_{N-1}} \Big[u'_{N-1}(B'QB + R)u_{N-1} + 2E\{x_{N-1} \mid I_{N-1}\}' A'QBu_{N-1} \Big]$$

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

$$u^*_{N-1} = \mu^*_{N-1}(I_{N-1}) = L_{N-1} E\{x_{N-1} \mid I_{N-1}\}$$

where

$$L_{N-1} = -(B'QB + R)^{-1} B'QA$$

DP ALGORITHM II

- Substituting in the DP algorithm

$$J_{N-1}(I_{N-1}) = \underset{x_{N-1}}{E}\left\{x'_{N-1}K_{N-1}x_{N-1} \mid I_{N-1}\right\}$$
$$+ \underset{x_{N-1}}{E}\Big\{\big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\big)'$$
$$\cdot P_{N-1}\big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\big) \mid I_{N-1}\Big\}$$
$$+ \underset{w_{N-1}}{E}\left\{w'_{N-1}Q_N w_{N-1}\right\},$$

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

$$P_{N-1} = A'_{N-1}Q_N B_{N-1}(R_{N-1} + B'_{N-1}Q_N B_{N-1})^{-1}$$
$$\cdot B'_{N-1}Q_N A_{N-1},$$
$$K_{N-1} = A'_{N-1}Q_N A_{N-1} - P_{N-1} + Q_{N-1}.$$

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

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

DP ALGORITHM III

- DP equation for period $N-2$:

$$J_{N-2}(I_{N-2}) = \min_{u_{N-2}} \Big[E_{x_{N-2},w_{N-2},z_{N-1}} \{x'_{N-2}Qx_{N-2} + u'_{N-2}Ru_{N-2} + J_{N-1}(I_{N-1}) \mid I_{N-2}, u_{N-2}\}\Big]$$

$$= E\Big\{x'_{N-2}Qx_{N-2} \mid I_{N-2}\Big\} + \min_{u_{N-2}} \Big[u'_{N-2}Ru_{N-2} + E\Big\{x'_{N-1}K_{N-1}x_{N-1} \mid I_{N-2}, u_{N-2}\Big\}\Big] + E\Big\{\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big)' \cdot P_{N-1}\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big) \mid I_{N-2}, u_{N-2}\Big\} + E_{w_{N-1}}\{w'_{N-1}Q_N w_{N-1}\}.$$

- 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

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

independently of the policy being used.

- The following simplified version of the lemma conveys the main idea.

- Simplified Lemma: Let r, u, z be random variables such that r and u are independent, and let $x = r + u$. Then

$$x - E\{x \mid z, u\} = r - E\{r \mid z\}.$$

- Proof: We have

$$\begin{aligned} x - E\{x \mid z, u\} &= r + u - E\{r + u \mid z, u\} \\ &= r + u - E\{r \mid z, u\} - u \\ &= r - E\{r \mid z, u\} \\ &= r - E\{r \mid z\}. \end{aligned}$$

APPLYING THE QUALITY OF ESTIMATION LEMM

- Using the lemma,

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

where

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

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

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

and is independent of u_{N-2}.

- So minimization in the DP algorithm yields

$$u^*_{N-2} = \mu^*_{N-2}(I_{N-2}) = L_{N-2} E\{x_{N-2} \mid I_{N-2}\}$$

FINAL RESULT

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

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

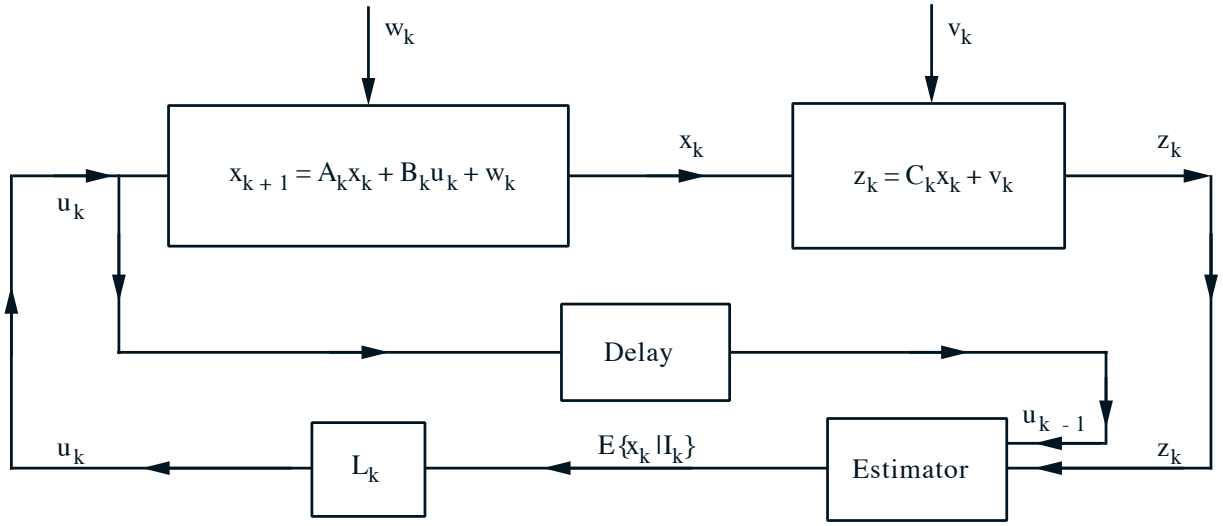
where L_k is the same as for perfect state info:

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

with K_k generated 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 $E\{x_k \mid I_k\}$.
  - (b) An *actuator*, which multiplies $E\{x_k \mid I_k\}$ by the gain matrix L_k and applies the control input $u_k = L_k E\{x_k \mid I_k\}$.

- Generically the estimate $\hat{x}$ of a random vector x given some information (random vector) I, which minimizes the mean squared error

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

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

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

- The actuator portion is optimal for the control problem assuming perfect state information.

STEADY STATE/IMPLEMENTATION ASPECTS

- As $N \to \infty$, the solution of the Riccati equation converges to a steady state and $L_k \to L$.

- If x_0, w_k, and v_k are Gaussian, $E\{x_k \mid I_k\}$ is a *linear* function of I_k and is generated by a nice recursive algorithm, the Kalman filter.

- The Kalman filter involves also a Riccati equation, so for $N \to \infty$, and a stationary system, it also has a steady-state structure.

- Thus, for Gaussian uncertainty, the solution is nice and possesses a steady state.

- For nonGaussian uncertainty, computing $E\{x_k \mid I_k\}$ maybe very difficult, so a suboptimal solution is typically used.

- Most common suboptimal controller: Replace $E\{x_k \mid I_k\}$ by the estimate produced by the Kalman filter (act as if x_0, w_k, and v_k are Gaussian).

- It can be shown that this controller is optimal within the class of controllers that are *linear* functions of I_k.

6.231 DYNAMIC PROGRAMMING

LECTURE 12

LECTURE OUTLINE

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

EVIEW: PROBLEM WITH IMPERFECT STATE INF

- 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:

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

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

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

subject to the equations

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

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

DP ALGORITHM

- DP algorithm:

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

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

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

- The optimal cost J^* is given by

$$J^* = \underset{z_0}{E}\Big\{ J_0(z_0) \Big\}.$$

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

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

- Such a function S_k is called a *sufficient statistic*.

- An optimal policy obtained by the preceding minimization can be written as

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

where $\overline{\mu}_k$ 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 $P_{X_K|I_K}$

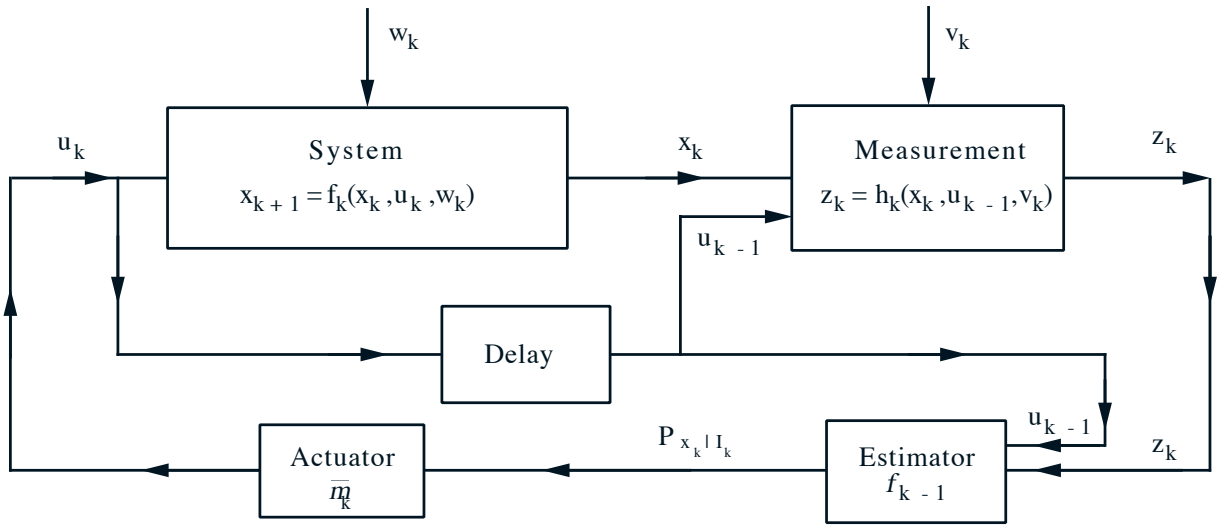
- It turns out that $P_{x_k|I_k}$ is generated recursively by a dynamic system (estimator) of the form

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

for a suitable function Φ_k

- DP algorithm can be written as

$$\overline{J}_k(P_{x_k|I_k}) = \min_{u_k \in U_k} \Big[\underset{x_k, w_k, z_{k+1}}{E} \big\{ g_k(x_k, u_k, w_k) + \overline{J}_{k+1}\big(\Phi_k(P_{x_k|I_k}, u_k, z_{k+1})\big) \mid I_k, u_k \big\} \Big]$$



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

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

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

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

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

- 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 $1, \ldots, n$.
- Then the conditional probability distribution $P_{x_k|I_k}$ is a vector

$$\big(P(x_k = 1 \mid I_k), \ldots, P(x_k = n \mid I_k)\big)$$

- 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 $\overline{J}_k$ in the DP algorithm are piecewise linear and concave (Exercise 5.7).
- This is conceptually important and also (moderately) useful in practice.

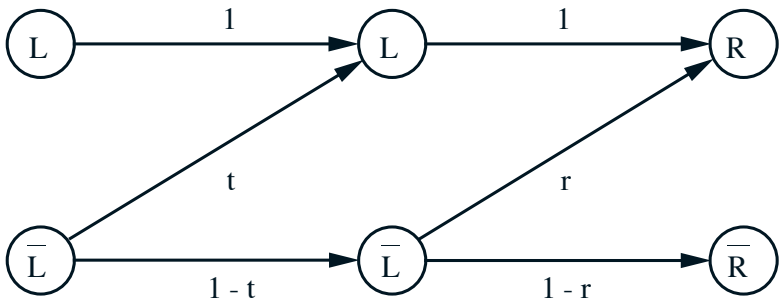
INSTRUCTION EXAMPLE

- Teaching a student some item. Possible states are L: Item learned, or $\overline{L}$: Item not learned.

- Possible decisions: T: Terminate the instruction, or $\overline{T}$: 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 $\overline{R}$: 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

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

- Using Bayes' rule we can obtain

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

- DP algorithm:

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

starting with

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

INSTRUCTION EXAMPLE III

- Write the DP algorithm as

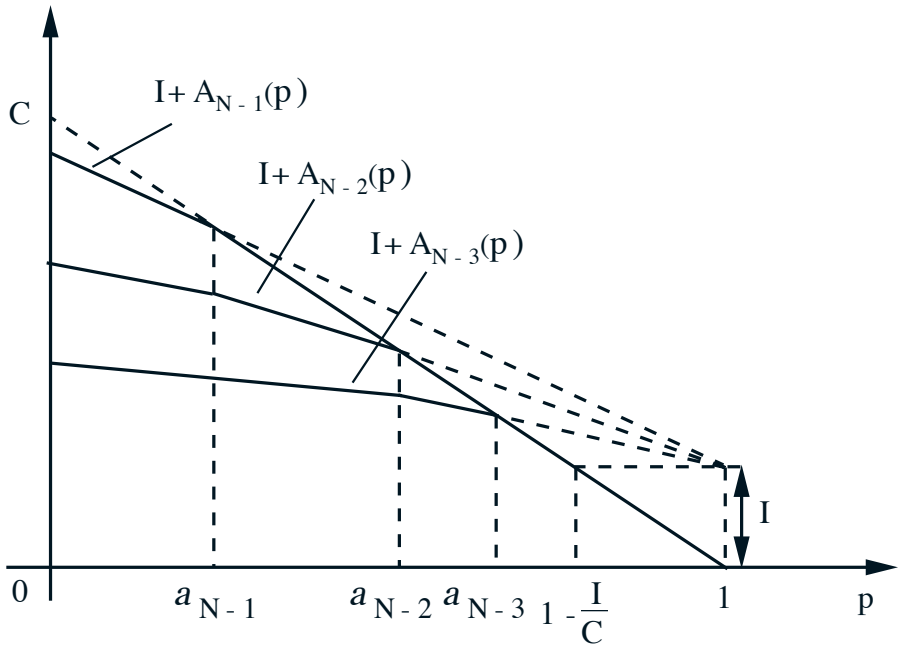
$$\overline{J}_k(p_k) = \min\big[(1-p_k)C,\, I + A_k(p_k)\big],$$

where

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

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

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



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 $\overline{x}_k(I_k)$ given the current information vector I_k.

(2) Fix the w_i, $i \geq k$, at some $\overline{w}_i(x_i, u_i)$. Solve the deterministic problem:

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

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

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

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

ALTERNATIVE IMPLEMENTATION

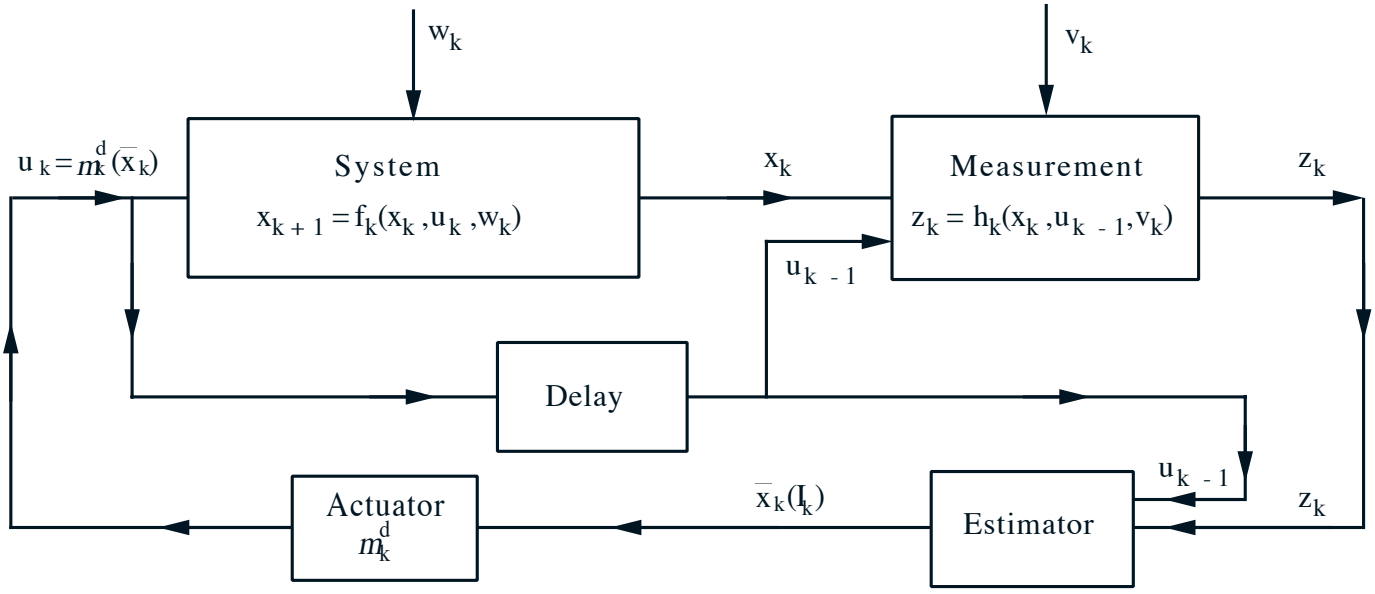
- Let $\{\mu_0^d(x_0), \ldots, \mu_{N-1}^d(x_{N-1})\}$ be an optimal controller obtained from the DP algorithm for the deterministic problem

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

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

The CEC applies at time k the control input

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



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 $u_k \in U_k(x_k)$

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

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 $\overline{x}_k(I_k)$ 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:

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

where $\overline{x}_k(I_k)$ 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 $\{\mu_0^*(I_0, \theta), \ldots, \mu_{N-1}^*(I_{N-1}, \theta)\}$ 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

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

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 control law μ^* that is optimal for known θ:

$$\hat{\mu}_k(I_k) = \mu_k^*(x_k, \hat{\theta}_k), \qquad \text{with } \hat{\theta}_k\text{: estimate of } \theta$$

There are three systems of interest:

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

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

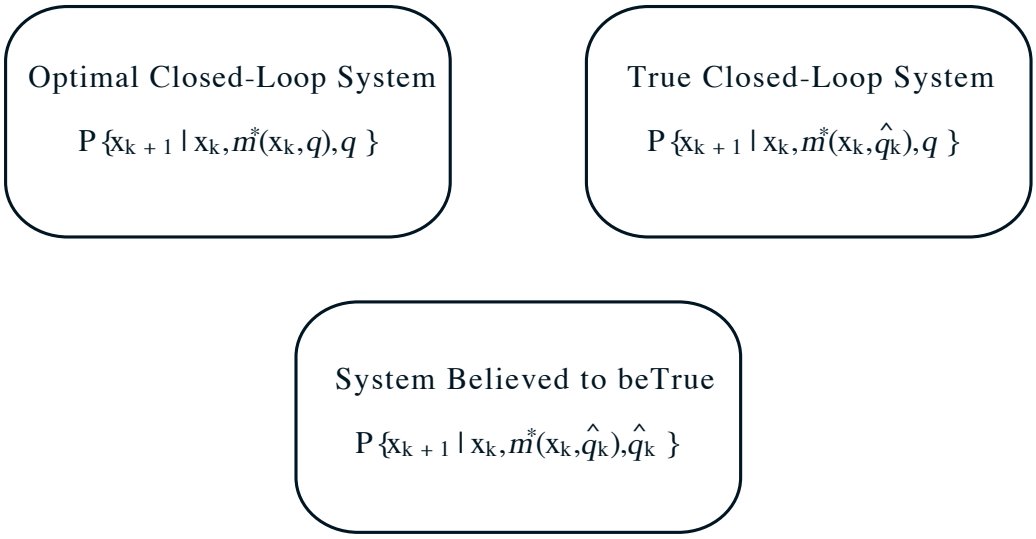
(b) The true closed-loop system, which evolves probabilistically according to

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

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

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

CEC AND IDENTIFIABILITY II



- **Difficulty:** 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,

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

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.

"REMEDY" TO IDENTIFIABILITY PROBLEM

- Introduce "noise" in the control applied, i.e., occasionally deviate from the CEC actions.
- This provides a means to escape from "wrong" estimates.
- However, introducing "noise" in the control may be difficult to implement in practice.
- Under some special circumstances, i.e., the "self-tuning" control context discussed in the book, the CEC is optimal in the limit, even if the parameter estimates converge to the wrong values.
- All of this touches upon some of the most sophisticated aspects of adaptive control.

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 $\overline{\mu}_k(x_k)$ that

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

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" $\overline{U}_k(x_k)$.

- As the length of lookahead increases, the required computation quickly explodes.

PERFORMANCE BOUNDS

- Let $\overline{J}_k(x_k)$ 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

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

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

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

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

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

- Important application: When $\tilde{J}_k$ is the cost-to-go of some heuristic policy (then the 1SL policy is called the *rollout* policy).

- The bound can be extended to the case where there is a δ_k in the RHS of (*). Then

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

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 kth 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 $r = (r_1, \ldots, r_m)$ 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.

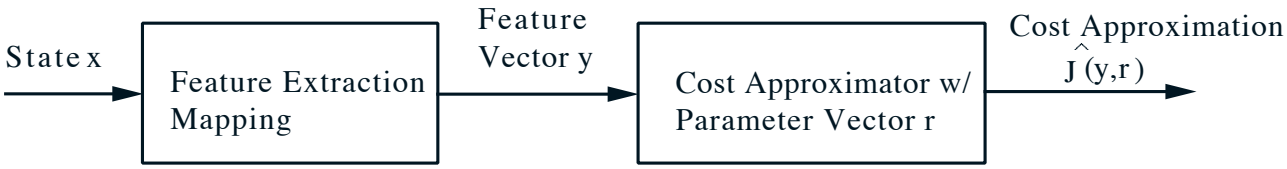
- Two key issues:
  - The choice of parametric class $\tilde{J}(x, r)$ (the approximation architecture).
  - Method for tuning the weights ("training" the architecture).

- 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

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

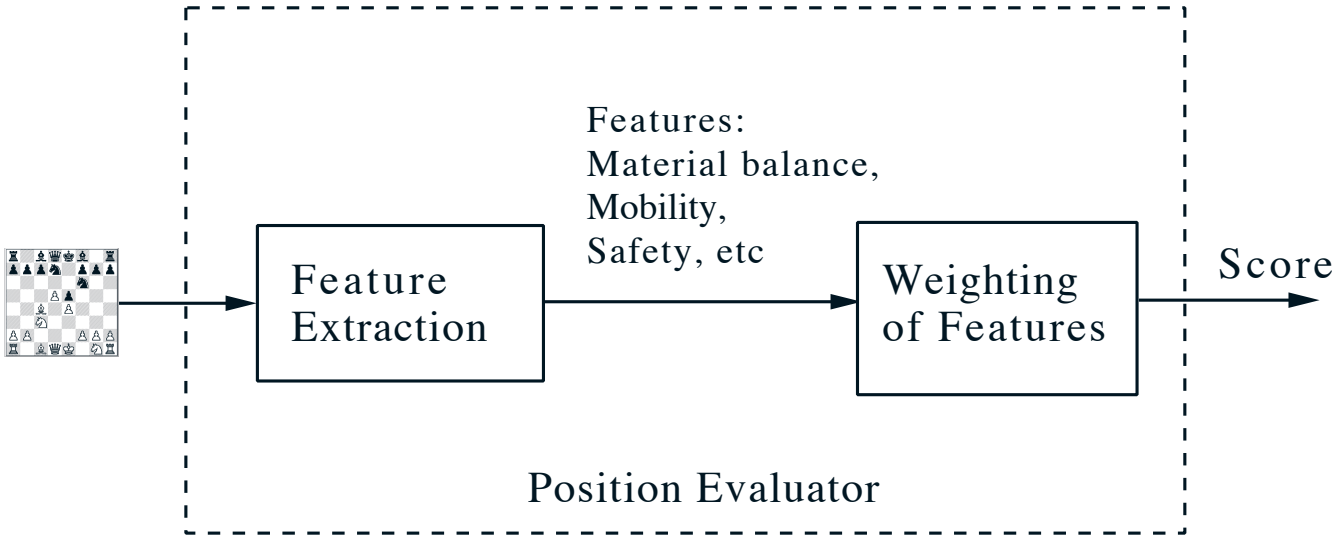


- Ideally, the features will encode much of the nonlinearity that is inherent in the cost-to-go approximated, and the approximation may be quite accurate without a complicated architecture.
- Sometimes the state space is partitioned, and "local" features are introduced for each subset of the partition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

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

COMPUTER CHESS I

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



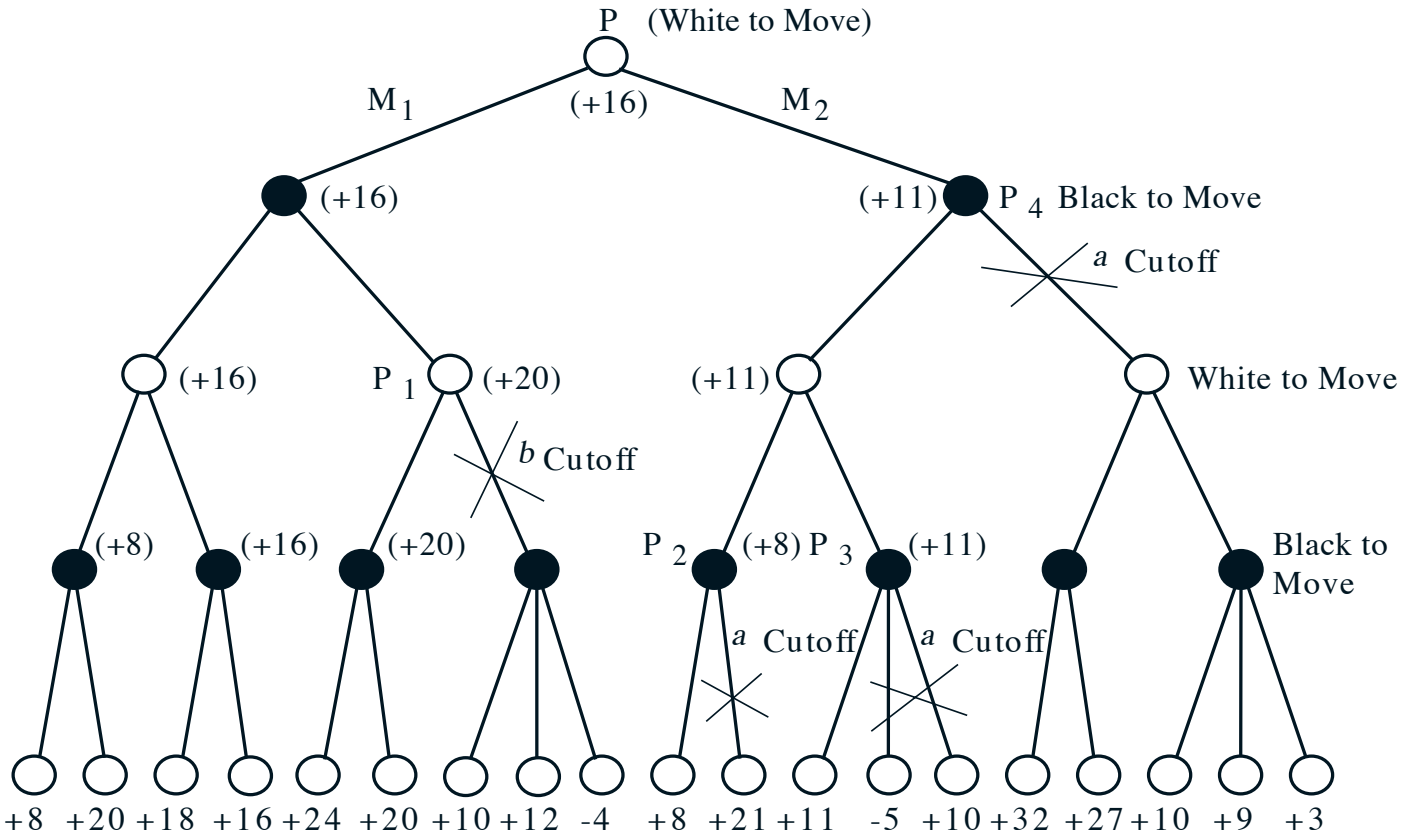
• Most often the weighting of features is linear but multistep lookahead is involved.

• 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 $\overline{\mu}_k(x_k)$ that

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

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

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

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

- We claim that $\overline{J}_k(x_k) \leq H_k(x_k)$ for all x_k and k

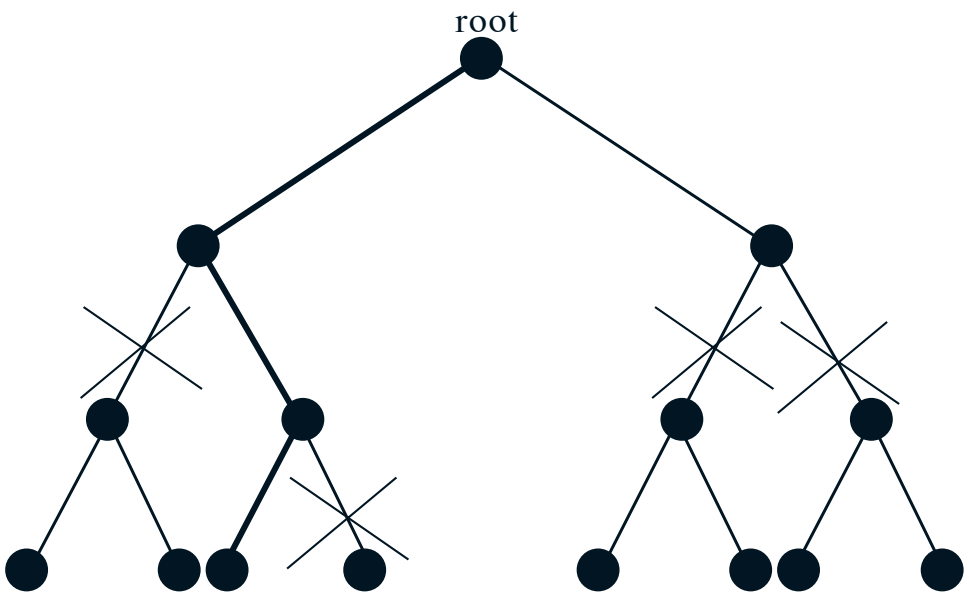
- Proof by induction: We have $\overline{J}_N(x_N) = H_N(x_N)$ for all x_N. Assume that

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

Then, for all x_k

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

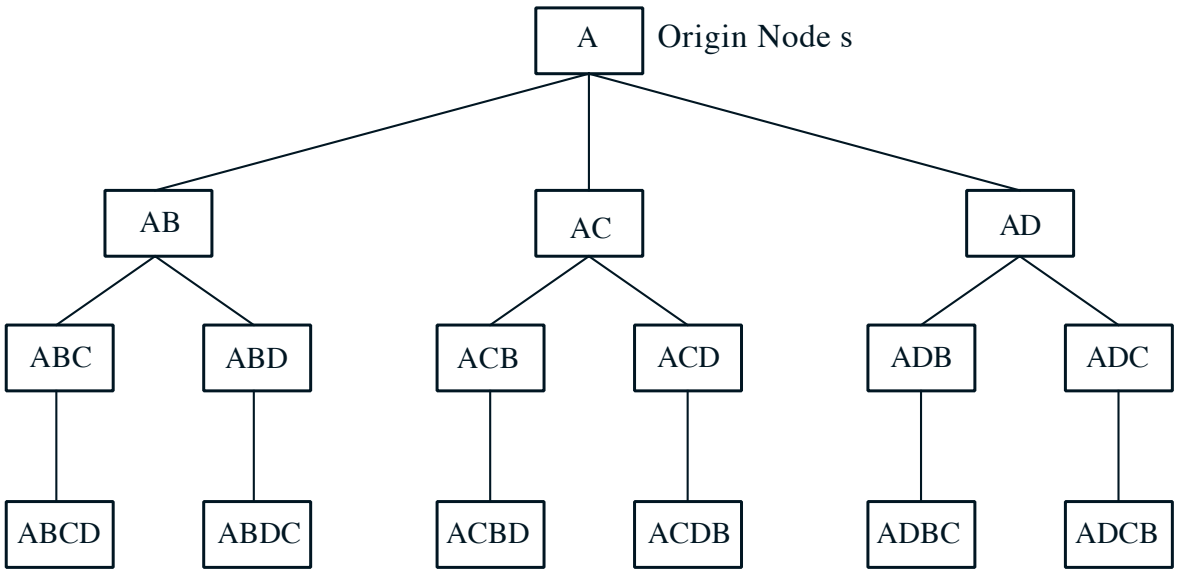
EXAMPLE: THE BREAKTHROUGH PROBLEM



- 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.



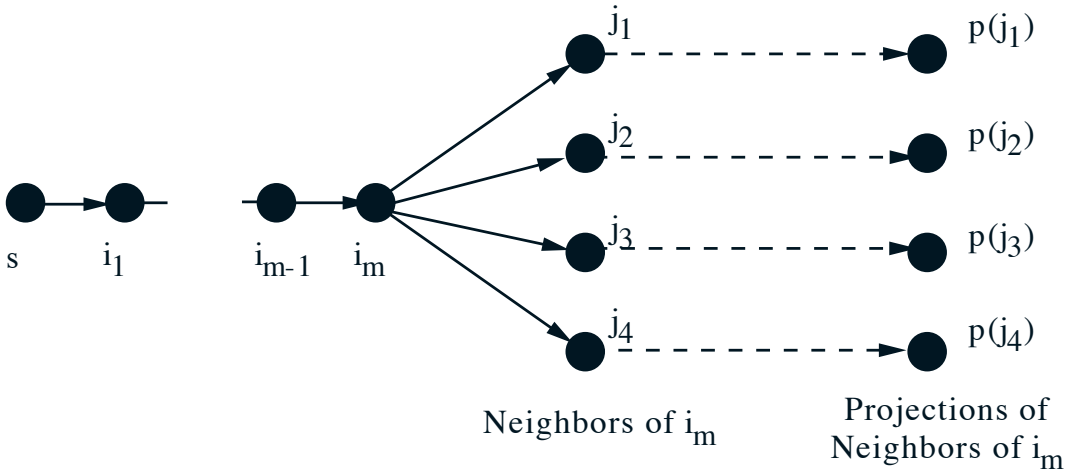
Traveling salesman problem with four cities A, B, C, D

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.

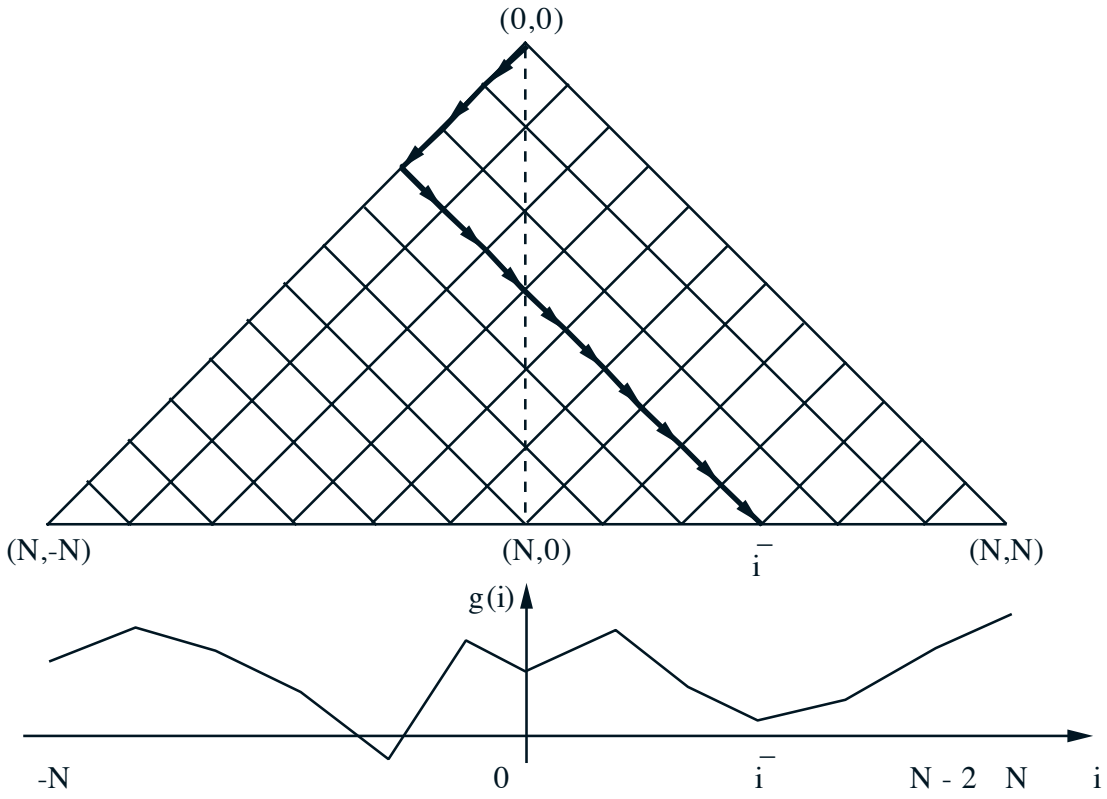
- Base heuristic: For any nondestination node i, constructs a path $(i, i_1, \ldots, i_m, \bar{i})$ starting at i and ending at one of the destination nodes $\bar{i}$. We call $\bar{i}$ the *projection* of i, and we denote $H(i) = g(\bar{i})$.

- 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 $(i, i_1, \ldots, i_m, \bar{i})$ starting at i, it also generates the path $(i_1, \ldots, i_m, \bar{i})$ 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 $(i, i_1, \ldots, i_m)$, 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 $(s, i_1, \ldots, i_{\bar{m}})$ is the rollout path, we have

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

where $H(i)$ = 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 $\overline{\mu}_k(x_k)$ that

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

where

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

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 $Q_k(x_k, u) - Q_k(x_k, \overline{u})$. This differencing often amplifies the simulation error in the calculation of the Q-factors.

- Potential remedy: Compare any two controls u and $\overline{u}$ by simulating $Q_k(x_k, u) - Q_k(x_k, \overline{u})$ 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 future disturbances at "typical" values $\overline{w}_{k+1}, \ldots, \overline{w}_{N-1}$ and approximate the Q-factors with

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

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

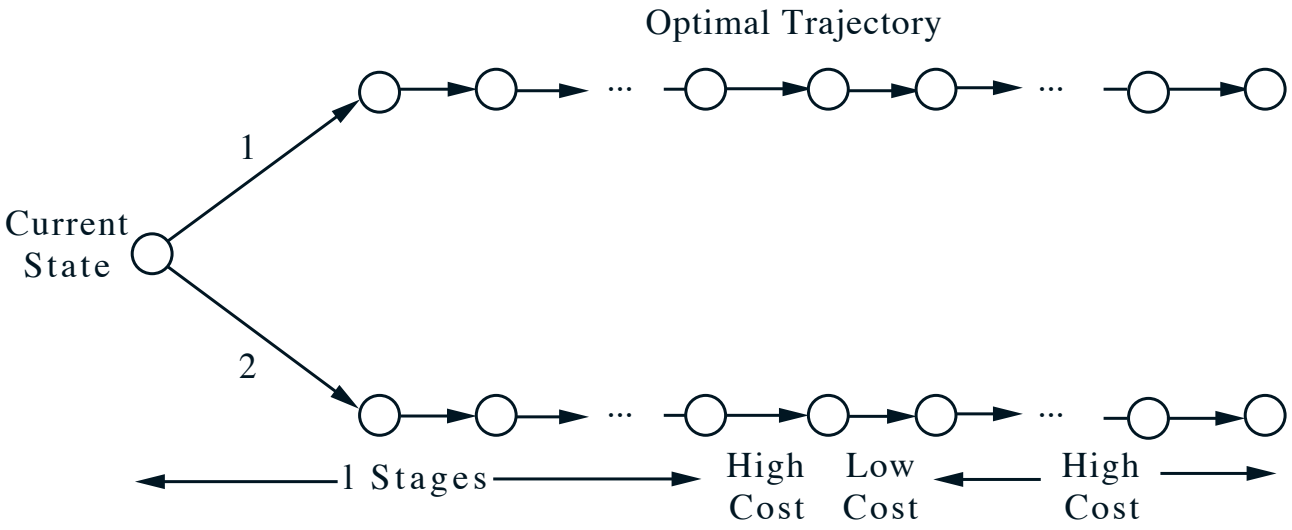
- This is an approximation of $H_{k+1}\Big(f_k(x_k, u_k, w_k)\Big)$ by using a "single sample simulation."

- Variant of the certainty equivalence approach: Approximate $H_{k+1}\Big(f_k(x_k, u_k, w_k)\Big)$ 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."

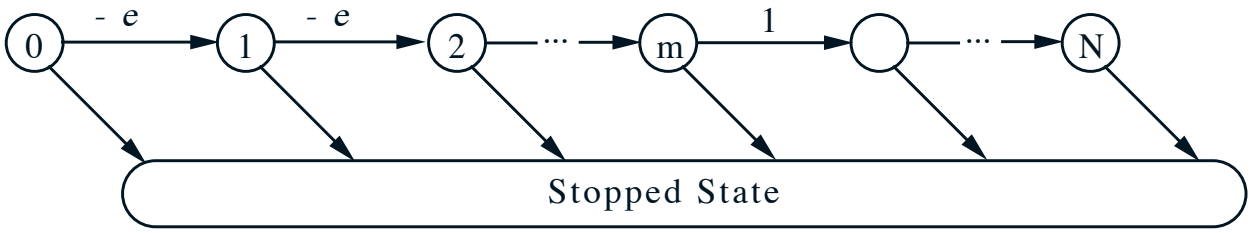
ROLLING HORIZON APPROACH

- 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:

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

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

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

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

- "Convexification effect" of continuous time.

GENERAL APPROACH FOR DISCRETIZATION I

- Given a discrete-time system with state space S, consider a finite subset $\overline{S}$; for example $\overline{S}$ could be a finite grid within a continuous state space S. Assume stationarity for convenience, i.e., that the system equation and cost per stage are the same for all times.

- We define an approximation to the original problem, with state space $\overline{S}$, as follows:

- Express each $x \in S$ as a convex combination of states in $\overline{S}$, i.e.,

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

- Define a "reduced" dynamic system with state space $\overline{S}$, whereby from each $x_i \in \overline{S}$ we move to $\overline{x} = f(x_i, u, w)$ according to the system equation of the original problem, and then move to $x_j \in \overline{S}$ with probabilities $\gamma_j(\overline{x})$.

- Define similarly the corresponding cost per stage of the transitions of the reduced system.

GENERAL APPROACH FOR DISCRETIZATION II

- Let $\overline{J}_k(x_i)$ be the optimal cost-to-go of the "reduced" problem from each state $x_i \in \overline{S}$ and time k onward.

- Approximate the optimal cost-to-go of any $x \in S$ for the original problem by

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

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

- The choice of coefficients $\gamma_i(x)$ is in principle arbitrary, but should aim at consistency, i.e., as the number of states in $\overline{S}$ increases, $\tilde{J}_k(x)$ should converge to the optimal cost-to-go of the original problem.

- Interesting observation: While the original problem may be deterministic, the reduced problem is always stochastic.

- Generalization: The set $\overline{S}$ may be any finite set (not a subset of $\overline{S}$) as long as the coefficients $\gamma_i(x)$ admit a meaningful interpretation that quantifies the degree of association of x with x_i.

OTHER SUBOPTIMAL CONTROL APPROACHES

- **Minimize the DP equation error:** 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.

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

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

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

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 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

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

  - 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

$$\lim_{N\to\infty} \frac{1}{N} \underset{\substack{w_k \\ k=0,1,\dots}}{E} \left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

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 $J_0(x) \equiv 0$

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

- Typical results for total cost problems:

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

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

(Bellman's Equation). If $\mu(x)$ minimizes in Bellman's Eq., the policy $\{\mu, \mu, \ldots\}$ 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 $1, \ldots, n$ and special cost-free termination state t
  - Transition probabilities $p_{ij}(u)$
  - Control constraints $u \in U(i)$
  - Cost of policy $\pi = \{\mu_0, \mu_1, \ldots\}$ is

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

  - Optimal policy if $J_\pi(i) = J^*(i)$ for all i.
  - Special notation: For stationary policies $\pi = \{\mu, \mu, \ldots\}$, 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 that m stages; for all π, we have

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

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

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

and similarly

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

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

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

and

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

MAIN RESULT

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

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

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

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

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

- 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,

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

vanishes as K increases to ∞.

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

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

$$J_\pi(x_0) = \sum_{k=0}^{mK-1} E\left\{g\big(x_k, \mu_k(x_k)\big)\right\} + \sum_{k=mK}^{\infty} E\left\{g\big(x_k, \mu_k(x_k)\big)\right\}$$

$$\leq \sum_{k=0}^{mK-1} E\left\{g\big(x_k, \mu_k(x_k)\big)\right\} + \sum_{k=K}^{\infty} \rho^k m \max_{i,u} |g(i,u)|$$

Take the minimum of both sides over π to obtain

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

Similarly, we have

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

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

- It can be seen that $J_{mK}(x_0)$ and $J_{mK+k}(x_0)$ converge to the same limit for $k = 1, \ldots, m-1$, so $J_N(x_0) \to J^*(x_0)$

EXAMPLE I

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

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

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

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

- 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

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

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 that 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 ($i = 1, \ldots, n$ and $t = 0$ the termination state).
- There is control choice only at state 1.

EXAMPLE II (CONTINUED)

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

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

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

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

with all other transition probabilities being 0.

- Bellman's equation:

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

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

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

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

  - Optimal policy if $J_\pi(i) = J^*(i)$ for all i.
  - Special notation: For stationary policies $\pi = \{\mu, \mu, \ldots\}$, 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 that m stages; for all π, we have

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

MAIN RESULT

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

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

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

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

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

- 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,

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

vanishes as K increases to ∞.

BELLMAN'S EQUATION FOR A SINGLE POLICY

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

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

- Proof: This is just Bellman's equation for a modified/restricted problem where there is only one policy, the stationary policy μ, i.e., the control constraint set at state i is $\tilde{U}(i) = \{\mu(i)\}$
- The equation provides a way to compute $J_\mu(i)$, $i = 1, \ldots, n$, but the computation is substantial for large n $[O(n^3)]$

POLICY ITERATION

- It generates a sequence $\mu^1, \mu^2, \ldots$ 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

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

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

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

- The algorithm stops when $J_{\mu^k}(i) = J_{\mu^{k+1}}(i)$ for all i

- Note the connection with the rollout algorithm, which is just a single policy iteration

JUSTIFICATION OF POLICY ITERATION

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

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

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

$$J_0(i) = g\big(i, \mu^k(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big) J_0(j)$$

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

Using the monotonicity property of DP,

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

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

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

LINEAR PROGRAMMING

• We claim that J^* is the "largest" J that satisfies the constraint

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

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

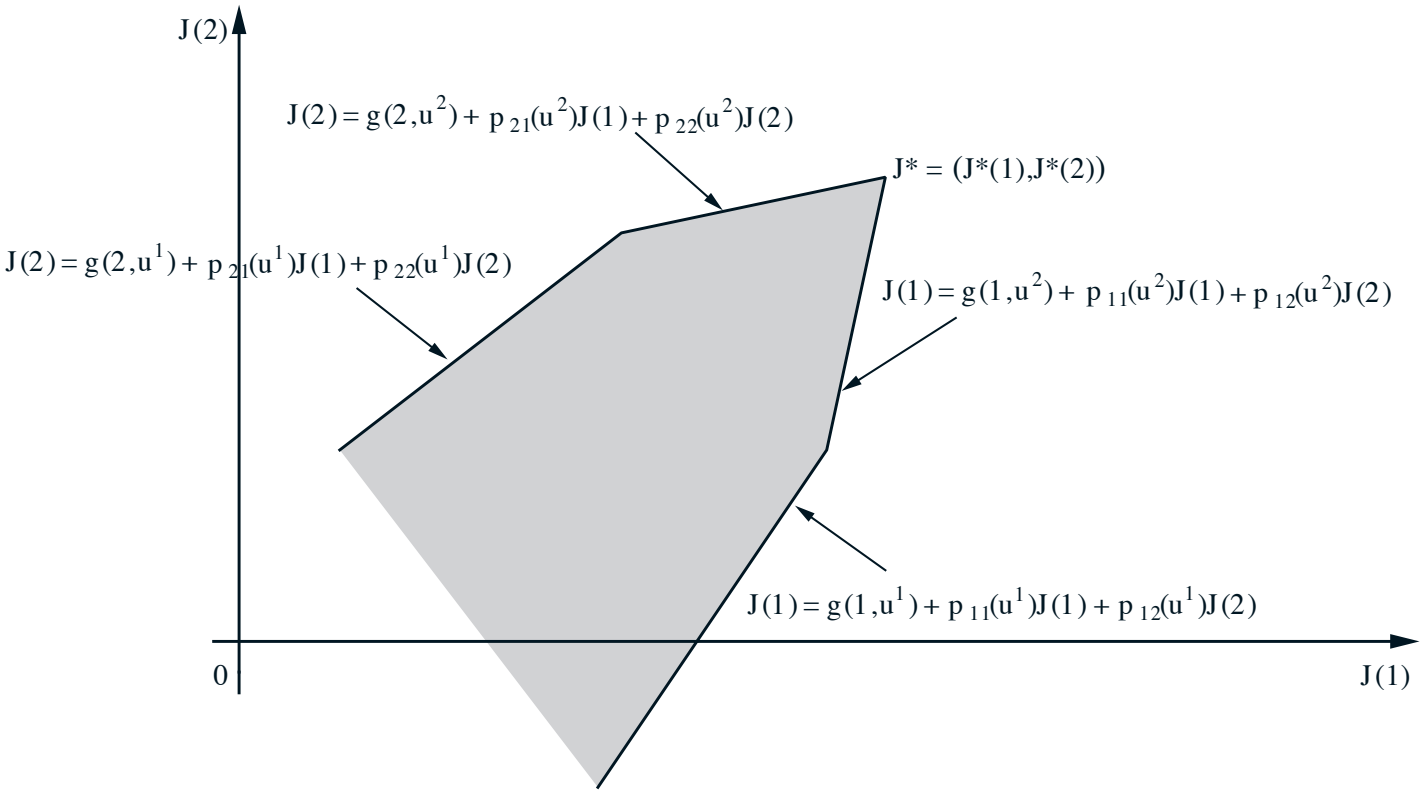
• Proof: If we use value iteration to generate a sequence of vectors $J_k = \big(J_k(1), \ldots, J_k(n)\big)$ starting with a J_0 such that

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

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

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

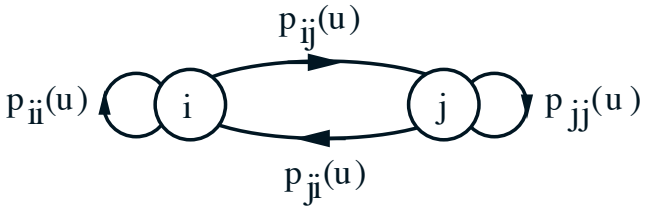
LINEAR PROGRAMMING (CONTINUED)

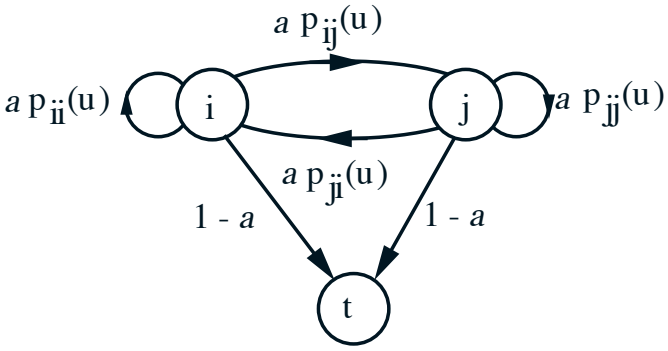


• Drawback: For large n the dimension of this program is very large. Furthermore, the number of constraints is equal to the number of state-control pairs.

DISCOUNTED PROBLEMS

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





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

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

- 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

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

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 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

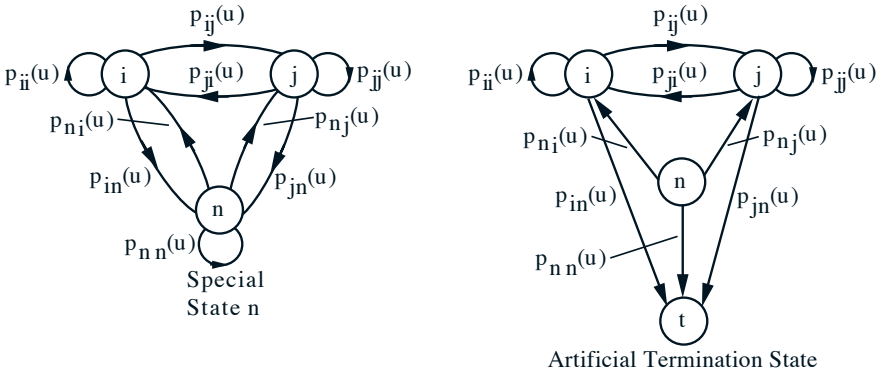
- Stationary system with finite number of states and controls
- Minimize over policies $\pi = \{\mu_0, \mu_1, ...\}$

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

- Important characteristics (not shared by other types of infinite horizon problems)
  - For any fixed K, the cost incurred up to time K does not matter (only the state that we are at time K matters)
  - If all states "communicate" the optimal cost is independent of 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 cost at i of the SSP be $g(i,u) - \lambda^*$
- We will show that

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

CONNECTION WITH SSP (CONTINUED)

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

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

where for a fixed μ,

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

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

- Intuitively, optimal cycle cost $= \lambda^*$, 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 $i = 1, \ldots, n$. Then, $h^*(1), \ldots, h^*(n)$ 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 WITH SSP

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

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

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

- We don't know λ^*, so we can't solve the average cost problem as an SSP problem. But similar value and policy iteration algorithms are possible.

- Example: A manufacturer at each time:
  - Receives an order with prob. p and no order with prob. $1-p$.
  - May process all unfilled orders at cost $K > 0$, or process no order at all. The cost per unfilled order at each time is $c > 0$.
  - Maximum number of orders that can remain unfilled is n.
  - Find a processing policy that minimizes the total expected cost per stage.

EXAMPLE (CONTINUED)

- State = number of unfilled orders. State 0 is the special state for the SSP formulation.
- Bellman's equation: For states $i = 0, 1, \ldots, n-1$

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

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:

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

- Result: $\lim_{k \to \infty} J_k(i)/k = \lambda^*$ for all i.

- Proof outline: Let J_k^* be so generated from the initial condition $J_0^* = h^*$. Then, by induction,

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

On the other hand,

$$\left| J_k(i) - J_k^*(i) \right| \leq \max_{j=1,\ldots,n} \left| J_0(j) - h^*(j) \right|, \qquad \forall \ i$$

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

RELATIVE VALUE ITERATION

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

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

- Then we can show $h_k \to h^*$ (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

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

- 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

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

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 kth 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

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

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

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

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

EXPONENTIAL TRANSITION DISTRIBUTIONS

- Important example of transition distributions

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

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

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

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

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

COST STRUCTURES

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

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

- 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 $\pi = \{\mu_0, \mu_1, \ldots\}$ starting from state i (with discount factor $\beta > 0$)

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

- Average cost per unit time

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

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

A NOTE ON NOTATION

- The scaled CDF $Q_{ij}(\tau, u)$ can be used to model discrete, continuous, and mixed distributions for the transition time τ.

- Generally, expected values of functions of τ can be written as integrals involving $d\,Q_{ij}(\tau, u)$. For example, the conditional expected value of τ given i, j, and u is written as

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

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

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

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

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

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

DISCOUNTED PROBLEMS – COST CALCULATION

- For a policy $\pi = \{\mu_0, \mu_1, \ldots\}$, 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 $\pi_1 = \{\mu_1, \mu_2, \ldots\}$

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

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

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

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

COST CALCULATION (CONTINUED)

- Also

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

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

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

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

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

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

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: For $i=1,\ldots,n$,

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

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

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

MANUFACTURER'S EXAMPLE REVISITED

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

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

- The one-stage expected cost G is

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

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

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

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

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

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

fill the orders <==> their number i exceeds i^*

AVERAGE COST

- Minimize

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

assuming there is a special state that is "recurrent under all policies"

- Total expected cost of a transition

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

where $\overline{\tau}_i(u)$: 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 $G(i,u) - \lambda^*\overline{\tau}_i(u)$, 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:

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

AVERAGE COST MANUFACTURER'S EXAMPLE

- The expected transition times are

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

the expected transition cost is

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

and there is also the "instantaneous" cost

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

- Bellman's equation:

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 21

LECTURE OUTLINE

- With this lecture, we start a four-lecture sequence on advanced dynamic programming and neuro-dynamic programming topics. References:
  - Dynamic Programming and Optimal Control, Vol. II, by D. Bertsekas
  - Neuro-Dynamic Programming, by D. Bertsekas and J. Tsitsiklis
- **1st Lecture:** Discounted problems with infinite state space, stochastic shortest path problem
- **2nd Lecture:** DP with cost function approximation
- **3rd Lecture:** Simulation-based policy and value iteration, temporal difference methods
- **4th Lecture:** Other approximation methods: Q-learning, state aggregation, approximate linear programming, approximation in policy space

DISCOUNTED PROBLEMS W/ BOUNDED COST

- System

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

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

$$J_\pi(x_0) = \lim_{N\to\infty} \underset{\substack{w_k \\ k=0,1,\ldots}}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

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)

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

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

- For any stationary policy μ

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

"SHORTHAND" THEORY

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

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

- Bellman's equation: $J^* = TJ^*, \quad J_\mu = T_\mu J_\mu$
- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

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

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

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

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

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

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

THE THREE KEY PROPERTIES

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

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

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

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

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

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

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

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

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

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

"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,

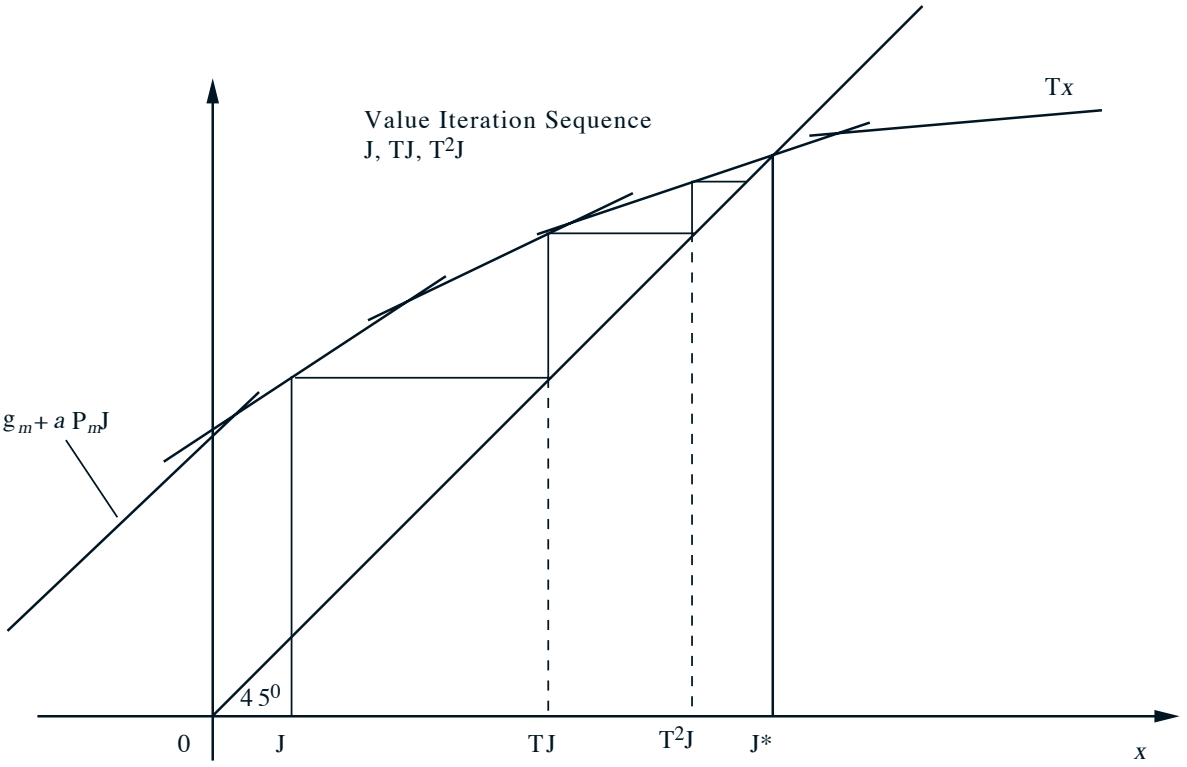
$$\max_x \left|(T^k J)(x) - J^*(x)\right| \leq \alpha^k \max_x \left|J(x) - J^*(x)\right|$$

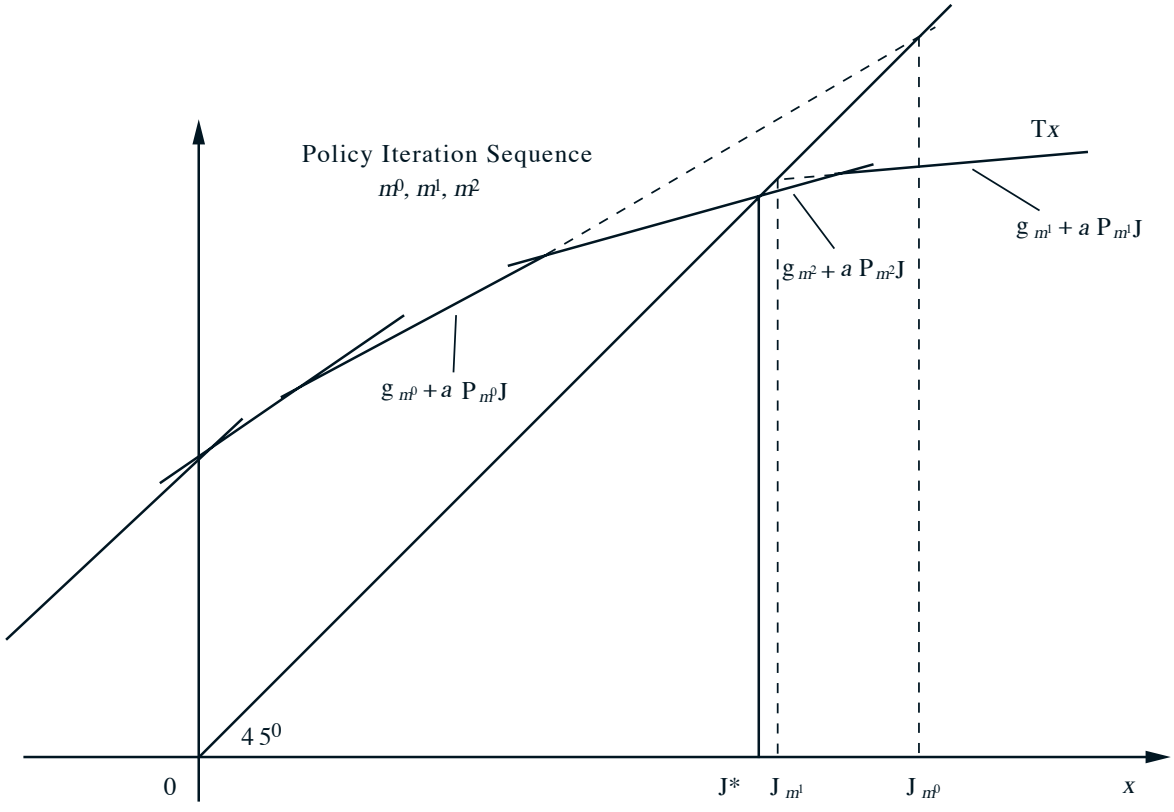
- An assortment of other analytical and computational results are based on the contraction property, e.g, error bounds, computational enhancements, etc.
- Example: If we execute value iteration *approximately*, so we compute TJ within an ϵ-error, i.e.,

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

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

GEOMETRIC INTERPRETATIONS





UNDISCOUNTED PROBLEMS

- System

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

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

$$J_\pi(x_0) = \lim_{N\to\infty} \mathop{E}_{\substack{w_k \\ k=0,1,\ldots}} \left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

- Shorthand notation for DP mappings

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

- For any stationary policy μ

$$(T_\mu J)(x) = \mathop{E}_{w} \left\{ g\big(x, \mu(x), w\big) + J\Big(f(x, \mu(x), w)\Big) \right\}, \ \forall\, x$$

- 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
- Mappings T and T_μ (modified to account for termination state t):

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

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

- **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

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

- If all μ are proper, then T is similarly a contraction (the case discussed in the text, 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

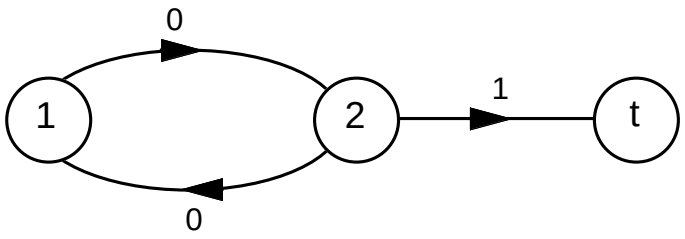
  (b) For each improper μ, $J_\mu(i) = \infty$ for some i

- Then T is not necessarily a contraction, but:
  - J^* is the unique solution of Bellman's Equ.
  - μ^* is optimal if and only if $T_{\mu^*} J^* = T J^*$
  - $\lim_{k \to \infty} (T^k J)(i) = J^*(i)$ for all i
  - Policy iteration terminates with an optimal policy, if started with a proper policy

- **Example:** Deterministic shortest path problem with a single destination t.
  - States <=> nodes; Controls <=> arcs
  - Termination state <=> the destination
  - Assumption (a) <=> every node is connected to the destination
  - Assumption (b) <=> all cycle costs > 0

ATHOLOGIES I: DETERMINISTIC SHORTEST PATH

- If there is a cycle with cost $= 0$, Bellman's equation has an infinite number of solutions. Example:



- We have $J^*(1) = J^*(2) = 1$.
- Bellman's equation is

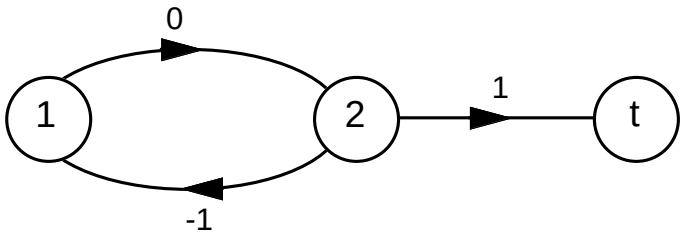
$$J(1) = J(2), \qquad J(2) = \min\big[J(1), 1\big].$$

- It has J^* as solution.
- Set of solutions of Bellman's equation:

$$\big\{J \mid J(1) = J(2) \leq 1\big\}.$$

ATHOLOGIES II: DETERMINISTIC SHORTEST PAT

- If there is a cycle with cost < 0, Bellman's equation has no solution [among functions J with $-\infty < J(i) < \infty$ for all i]. Example:



- We have $J^*(1) = J^*(2) = -\infty$.
- Bellman's equation is

$$J(1) = J(2), \qquad J(2) = \min\big[-1 + J(1), 1\big].$$

- There is no solution [among functions J with $-\infty < J(i) < \infty$ for all i].
- Bellman's equation has as solution $J^*(1) = J^*(2) = -\infty$ [within the larger class of functions $J(\cdot)$ that can take the value $-\infty$ for some (or all) states]. This observation can be generalized (see Chapter 3 of Vol. II of the text).

ATHOLOGIES III: THE BLACKMAILER'S DILEMM

- 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 in 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.)

6.231 DYNAMIC PROGRAMMING

LECTURE 22

LECTURE OUTLINE

- Approximate DP for large/intractable problems
- Approximate policy iteration
- Simulation-based policy iteration
- Actor-critic interpretation
- Learning how to play tetris: A case study
- Approximate value iteration with function approximation

PPROX. POLICY ITERATION - DISCOUNTED CAS

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

$$\max_x |J_k(x) - J_{\mu^k}(x)| \leq \delta, \qquad k = 0, 1, \ldots$$

and policy improvement is also approximate, according to,

$$\max_x |(T_{\mu^{k+1}} J_k)(x) - (T J_k)(x)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

where δ and ϵ are some positive scalars.

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

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

- 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^*.

APPROXIMATE POLICY ITERATION - SSP

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

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

and policy improvement is also approximate, according to,

$$\max_{i=1,\ldots,n} |(T_{\mu^{k+1}} J_k)(i) - (TJ_k)(i)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

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

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

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

SIMULATION-BASED POLICY EVALUATION

- Given μ, suppose we want to calculate J_μ by simulation.

- Generate by simulation sample costs. Approximation:

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

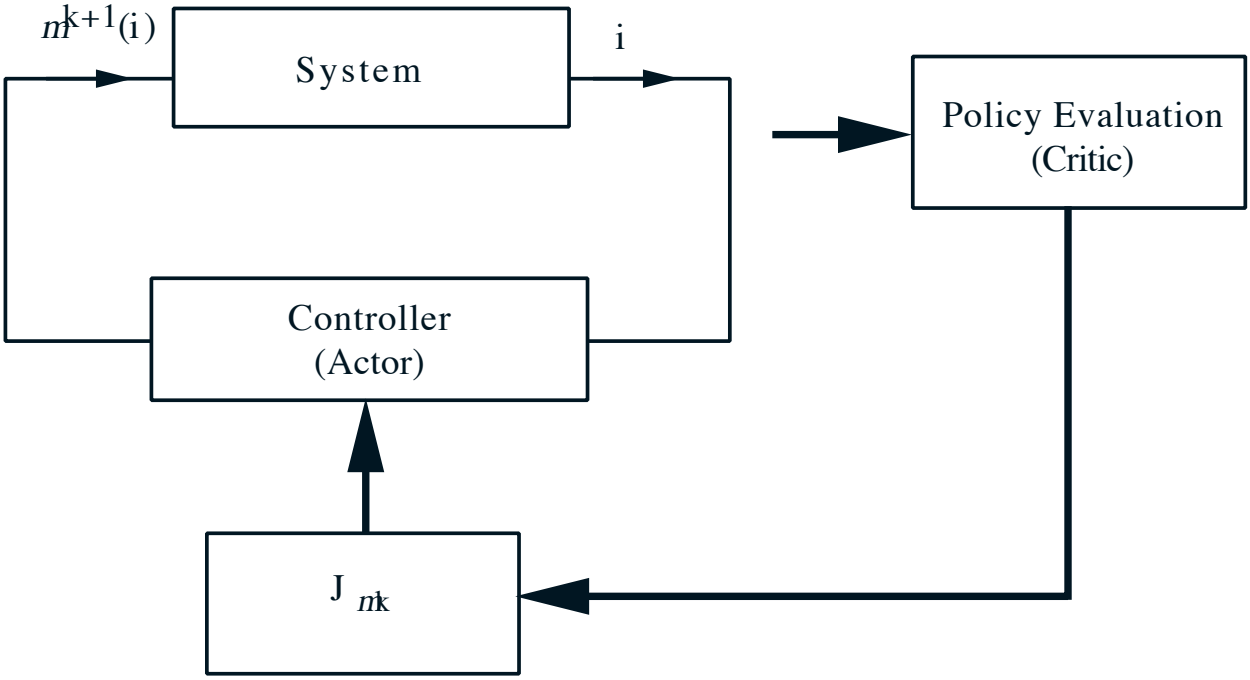
$c(i,m)$: mth sample cost starting from state i

- 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

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

- This idea is the starting point for more sophisticated simulation-related methods, to be discussed in the next lecture.

ACTOR-CRITIC INTERPRETATION

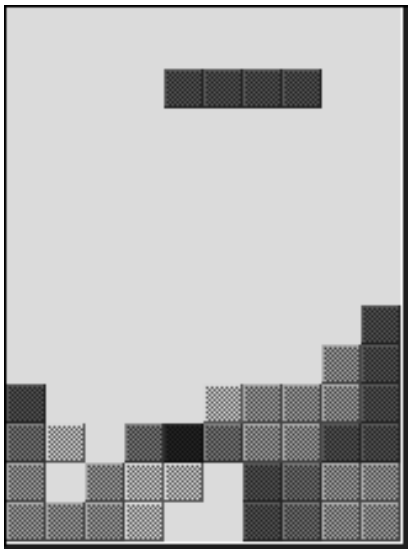


• The critic calculates approximately (e.g., using some form of a least squares fit) J_{μ^k} by processing state/sample cost pairs, which are generated by the actor by simulation

• Given the approximate J_{μ^k}, the actor implements the improved policy $J_{\mu^{k+1}}$ by

$$(T_{\mu^{k+1}} J_k)(i) = (T J_k)(i)$$

EXAMPLE: TETRIS I



- The state consists of the board position i, and the shape of the current falling block (astronomically large number of states).
- It can be shown that all policies are proper!!
- Use a linear approximation architecture with feature extraction

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

where $r = (r_1, \ldots, r_s)$ is the parameter vector and $\phi_m(i)$ is the value of mth feature associated w/ i.

EXAMPLE: TETRIS II

- Approximate policy iteration was implemented with the following features:
  - The height of each column of the wall
  - The difference of heights of adjacent columns
  - The maximum height over all wall columns
  - The number of "holes" on the wall
  - The number 1 (provides a constant offset)
- Playing data was collected for a fixed value of the parameter vector r (and the corresponding policy); the policy was approximately evaluated by choosing r to match the playing data in some least-squares sense.
- The method used for approximate policy evaluation was the λ-*least squares policy evaluation method*, to be described in the next lecture.
- See: Bertsekas and Ioffe, "Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming," in

http://www.mit.edu:8001//people/dimitrib/publ.html

ALUE ITERATION W/ FUNCTION APPROXIMATIO

- Suppose we use a linear approximation architecture $\tilde{J}(i,r) = \phi(i)'r$, or

$$\tilde{J} = \Phi r$$

where $r = (r_1, \ldots, r_s)$ is a parameter vector, and Φ is a full rank $n \times s$ feature matrix.

- **Approximate value iteration method:** Start with initial guess r_0; given r_t, generate r_{t+1} by

$$r_{t+1} = \arg\min_r \left\| \Phi r - T(\Phi r_t) \right\|$$

where $\|\cdot\|$ is some norm.

- Questions: Does r_t converge to some r^*? How close is Φr^* to J^*?

- **Convergence Result:** If T is a contraction with respect to a weighted Euclidean norm ($\|J\|^2 = J'DJ$, where D is positive definite, symmetric), then r_t converges to (the unique) r^* satisfying

$$r^* = \arg\min_r \left\| \Phi r - T(\Phi r^*) \right\|$$

GEOMETRIC INTERPRETATION

- Consider the feature subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

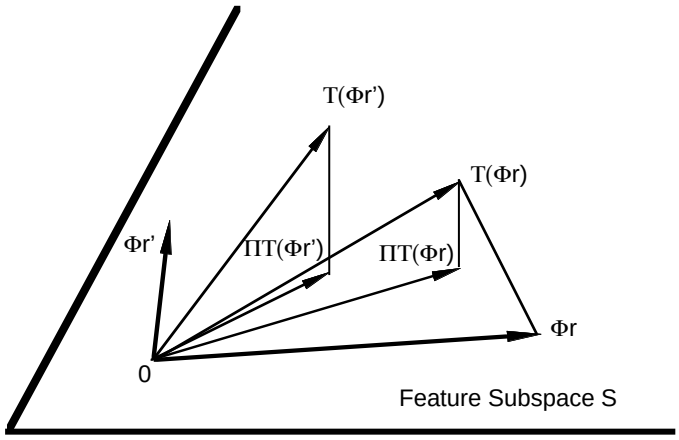
of all cost function approximations that are linear combinations of the feature vectors. Let Π denote projection on this subspace.

- The approximate value iteration is

$$r_{t+1} = \Pi T(\Phi r_t) = \arg\min_r \big\|\Phi r - T(\Phi r_t)\big\|$$

and amounts to starting at the point Φr_t of S applying T to it and then projecting on S.

- **Proof Idea:** Since T is a contraction with respect to the norm of projection, and projection is nonexpansive, ΠT (which maps S to S) is a contraction (with respect to the same norm).



PROOF

• Consider two vectors Φr and $\Phi r'$ in S. The (Euclidean) projection is a nonexpansive mapping, so

$$\|\Pi T(\Phi r) - \Pi T(\Phi r')\| \leq \|T(\Phi r) - T(\Phi r')\|$$

Since T is a contraction mapping (with respect to the norm of projection),

$$\|T(\Phi r) - T(\Phi r')\| \leq \beta \|\Phi r - \Phi r'\|$$

where $\beta \in (0, 1)$ is the contraction modulus, so

$$\|\Pi T(\Phi r) - \Pi T(\Phi r')\| \leq \beta \|\Phi r - \Phi r'\|$$

and it follows that ΠT is a contraction (with respect to the same norm and with the same modulus).

• In general, it is not clear how to obtain a Euclidean norm for which T is a contraction.

• **Important fact:** In the case where $T = T_\mu$, where μ is a stationary policy, T is a contraction for the norm $\|J\|^2 = J'DJ$, where D is diagonal with the steady-state probabilities along the diagonal.

ERROR BOUND

• If T is a contraction with respect to a weighted Euclidean norm $\|\cdot\|$ with modulus β, and r^* is the limit of r_t, i.e.,

$$r^* = \arg\min_r \big\|\Phi r - T(\Phi r^*)\big\|$$

then

$$\|\Phi r^* - J^*\| \leq \frac{\|\Pi J^* - J^*\|}{1-\beta}$$

where J^* is the fixed point of T, and ΠJ^* is the projection of J^* on the feature subspace S (with respect to norm $\|\cdot\|$).

Proof: Using the triangle inequality,

$$\begin{aligned}\|\Phi r^* - J^*\| &\leq \|\Phi r^* - \Pi J^*\| + \|\Pi J^* - J^*\| \\ &= \|\Pi T(\Phi r^*) - \Pi T(J^*)\| + \|\Pi J^* - J^*\| \\ &\leq \beta\|\Phi r^* - J^*\| + \|\Pi J^* - J^*\| \qquad \text{Q.E.D.}\end{aligned}$$

• Note that the error $\|\Phi r^* - J^*\|$ is proportional to $\|\Pi J^* - J^*\|$, which can be viewed as the "power of the approximation architecture" (measures how well J^* can be represented by the chosen features).

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

- Simulation-based policy and value iteration methods
- λ-Least Squares Policy Evaluation method
- Temporal differences implementation
- Policy evaluation by approximate value iteration
- TD(λ)

POLICY AND VALUE ITERATION BY SIMULATION

- There are many proposals, but we will focus on methods for which there is solid theory:

(a) **Policy evaluation** methods, to be used in exact or approximate policy iteration.

- Here the policy is fixed.
- As a special case we obtain the rollout method.
- 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 (3) Through on-line simulation as needed (rollout algorithm).

(b) **Value iteration** w/ function approximation.

- A big restriction is to find a suitable Euclidean norm for which T is a contraction.
- Such a norm can be found in the case where there is only one policy ($T = T_\mu$).
- Q-Learning is a form of on-line simulation-based value iteration method, but the only available theory applies to the lookup table representation case.

SIMULATION-BASED POLICY EVALUATION

- The policy is fixed and one or more long 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).
- For on-line methods, a sequence $\{r_t\}$ of parameter vectors is generated.
- 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 that solves a related least-squares approximation problem.
- We will focus on so-called *temporal difference methods*, λ-least squares and TD(λ), which may be viewed as on-line simulation-based approximate value iteration methods for policy evaluation.

POLICY EVALUATION BY VALUE ITERATION I

- The remainder of this lecture is based on the paper "Improved Temporal Difference Methods with Function Approximation," by Bertsekas, Borkar, and Nedic at

http://www.mit.edu:8001//people/dimitrib/publ.html

- Let J be the cost function associated with a stationary policy in the discounted context, so J is the unique solution of Bellman's Eq., $J(i) = \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha J(j)\big) \equiv (TJ)(i)$. We assume that the associated Markov chain has steady-state probabilities $\overline{p}(i)$ which are all positive.

- If we use a linear approximation architecture $\tilde{J}(i,r) = \phi(i)'r$, the value iteration

$$J_{t+1}(i) = \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha J_t(j)\big) = (TJ_t)(i)$$

is approximated as $\Phi r_{t+1} \approx T(\Phi r_t)$ in the sense

$$r_{t+1} = \arg\min_{r} \sum_{i=1}^{n} w(i) \left(\phi(i)'r - \sum_{j=1}^{n} p_{ij}\Big(g(i,j) + \alpha\phi(j)'r_t\Big) \right)^2$$

where the $w(i)$ are some positive weights.

POLICY EVALUATION BY VALUE ITERATION II

- Assuming Φ has full rank, r_{t+1} is uniquely obtained by projecting the value iterate $T(\Phi r_t) = P(g + \alpha\Phi r_t)$ on the range space of Φ, where projection is with respect to the norm $\|z\|_D = \sqrt{z'Dz}$, and D is diagonal with the $w(i)$ along the diagonal.

- The iteration converges if the mapping T is a contraction with respect to the norm $\|\cdot\|_D$.
Key fact: This is so if the $w(i)$ are equal to the steady state probabilities $\overline{p}(i)$. The limit is the unique r^* satisfying

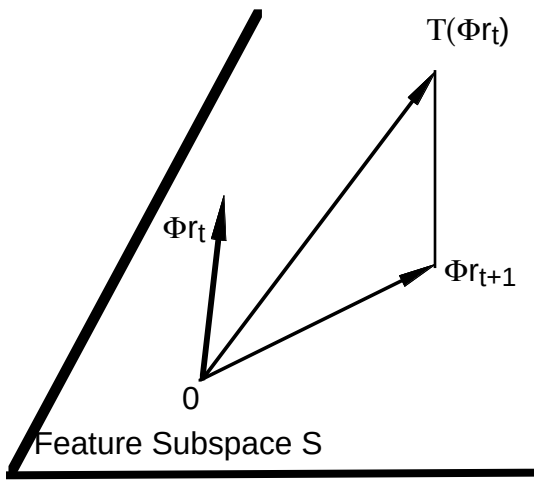
$$r^* = \arg\min_r \sum_{i=1}^{n} \overline{p}(i) \left(\phi(i)'r - \sum_{j=1}^{n} p_{ij} \left(g(i,j) + \alpha\phi(j)'r^* \right) \right)^2$$

- **Simulation-based implementation:** Generate an infinitely long trajectory $(i_0, i_1, \ldots)$ using a simulator, and iteratively update r by

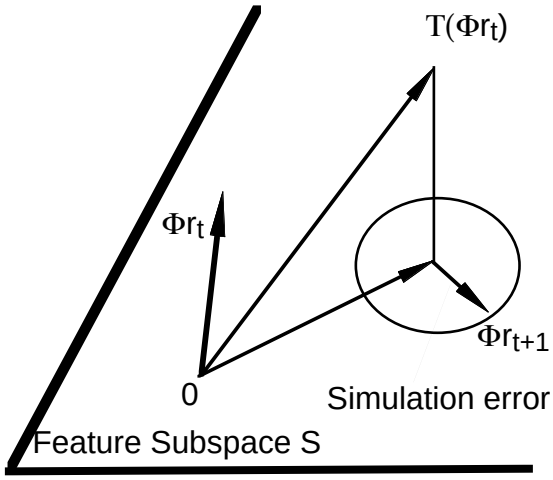
$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \underbrace{\left(\phi(i_m)'r - g(i_m, i_{m+1}) - \alpha\phi(i_{m+1})'r_t \right)^2}_{\text{squared value iteration error at time } m}$$

This can be shown to converge to the same r^*.

GEOMETRIC INTERPRETATION



Value Iteration with Linear Function Approximation



Simulation-Based Value Iteration with Linear Function Approximation

- The simulation-based implementation yields the (non-simulation) value iterate with linear function approximation [i.e., the projection of $T(\Phi r_t)$] plus stochastic simulation error.

- **Key Convergence Proof Idea:** The simulation error converges to 0 as the simulation trajectory becomes longer. Furthermore, the (non-simulation) value iteration is a convergent linear deterministic algorithm [since it involves a contraction mapping with respect to the weighted norm defined by the steady-state probabilities $\overline{p}(i)$].

USING M-STEP VALUE ITERATION

- For $M \geq 1$, consider the equation

$$J(i) = E\left[\alpha^M J(i_M) + \sum_{k=0}^{M-1} \alpha^k g(i_k, i_{k+1}) \;\middle|\; i_0 = i\right]$$

- This is Bellman's Eq. for a modified problem, involving a Markov chain where each transition corresponds to M transitions of the original, and the cost is calculated using a discount factor α^M and a cost per stage equal to $\sum_{k=0}^{M-1} \alpha^k g(i_k, i_{k+1})$.

- This Bellman equation is also solved uniquely by the same J that solves the ordinary (one-step) Bellman equation $J(i) = E\big[g(i,j) + \alpha J(j)\big]$.

- The corresponding value iteration method is

$$J_{t+1}(i) = E\left[\alpha^M J_t(i_M) + \sum_{k=0}^{M-1} \alpha^k g(i_k, i_{k+1}) \;\middle|\; i_0 = i\right]$$

and can be similarly approximated by simulation.

SIMULATION-BASED M-STEP VALUE ITERATION

- The corresponding simulation-based least-squares implementation is

$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \Bigg(\phi(i_m)'r - \alpha^M \phi(i_{m+M})' r_t - \underbrace{\sum_{k=0}^{M-1} \alpha^k g(i_{m+k}, i_{m+k+1}) \Bigg)^2}_{\text{squared } M\text{-step value iteration error}}$$

- Introducing the temporal differences, defined by

$$d_t(i_k, i_{k+1}) = g(i_k, i_{k+1}) + \alpha\phi(i_{k+1})'r_t - \phi(i_k)'r_t,$$

we can write this iteration as

$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \Bigg(\phi(i_m)'r - \phi(i_m)'r_t - \sum_{k=m}^{m+M-1} \alpha^{k-m} d_t(i_k, i_{k+1}) \Bigg)^2$$

USING RANDOM STEP VALUE ITERATION

- Consider a version of Bellman's equation where M is random and geometrically distributed with parameter λ, i.e.,

$$\text{Prob}(M = m) = (1 - \lambda)\lambda^{m-1}, \qquad m = 1, 2, \ldots$$

- This equation is obtained by multiplying both sides of the M-step Bellman's Eq. with $(1-\lambda)\lambda^{m-1}$, for each m, and adding over m:

$$J(i) = \sum_{m=1}^{\infty} (1-\lambda)\lambda^{m-1} E\left[\alpha^m J(i_m) + \sum_{k=0}^{m-1} \alpha^k g(i_k, i_{k+1}) \mid i_0 = i\right]$$

- The corresponding value iteration method is

$$J_{t+1}(i) = \sum_{m=1}^{\infty} (1-\lambda)\lambda^{m-1} E\left[\alpha^m J_t(i_m) + \sum_{k=0}^{m-1} \alpha^k g(i_k, i_{k+1}) \mid i_0 = i\right]$$

TEMPORAL DIFFERENCES IMPLEMENTATION

- We can write the random step value iteration as

$$J_{t+1}(i) = J_t(i) + \sum_{k=0}^{\infty} (\alpha\lambda)^k E\Big[g(i_k, i_{k+1}) + \alpha J_t(i_{k+1}) - J_t(i_k) \mid i_0 = i\Big]$$

- By using $\phi(i)'r_t$ to approximate J_t, and by replacing $g(i_k, i_{k+1}) + \alpha J_t(i_{k+1}) - J_t(i_k)$ with the temporal differences (TD)

$$d_t(i_k, i_{k+1}) = g(i_k, i_{k+1}) + \alpha\phi(i_{k+1})'r_t - \phi(i_k)'r_t,$$

we obtain the simulation-based least-squares implementation (called λ-least squares policy evaluation method)

$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \left(\phi(i_m)'r - \phi(i_m)'r_t - \sum_{k=m}^{t} (\alpha\lambda)^{k-m} d_t(i_k, i_{k+1}) \right)^2$$

- Role of the TD: They simplify the formulas.

- Convergence can be shown to an r^* that solves a corresponding least squares problem.

TD(LAMBDA)

• Another method for solving the policy evaluation problem is TD(λ), which uses a parameter $\lambda \in [0, 1]$ and generates an infinitely long trajectory $(i_0, i_1, \ldots)$ using a simulator. It iteratively updates r by

$$r_{t+1} = r_t + \gamma_t \left(\sum_{m=0}^{t} (\alpha\lambda)^{t-m} \phi(i_m) \right) d_t(i_t, i_{t+1})$$

where γ_t is a positive stepsize with $\gamma_t \to 0$.

• It can be viewed as a gradient-like method for minimizing the least-squares sum of the preceding λ-least squares method described earlier (see the Bertsekas, Borkar, and Nedic paper).

• For a given value of $\lambda \in [0, 1]$, TD(λ) converges to the same limit as the λ-least squares method (under technical assumptions on the choice of γ_t).

• While TD(λ) uses a simpler formula, it tends to be much slower than λ-Least Squares. In practice, it also requires tricky trial and error to settle on good stepsize choices.

TD METHODS: PROPERTIES AND DIFFICULTIES

- As M increases, the M-step Bellman's equation becomes better suited for approximation, because it embodies a longer horizon cost. Thus Φr^* tends to be closer to J when M is large.

- Similarly, Φr^* tends to be closer to J as $\lambda \approx 1$.

- On the other hand, when M or λ is large, the simulation noise inherent in the updates is magnified (more random cost terms are added), and convergence can be very slow. TD(λ) is particularly susceptible to noise, so $\lambda \approx 1$ may be a bad choice. This is less of a problem for the alternative λ-least squares method.

- A serious problem arises when the Markov chain is "slow-mixing," i.e., it takes many transitions for the simulation to reach important parts of the state space. Then if the simulation trajectory is terminated prematurely, the approximation obtained over these parts will be poor. A remedy is to use many long simulation trajectories starting from a set of initial states that adequately covers the state space.

6.231 DYNAMIC PROGRAMMING

LECTURE 24

LECTURE OUTLINE

- Additional methods for approximate DP
- Q-Learning
- Aggregation
- Linear programming with function approximation
- Gradient-based approximation in policy space

Q-LEARNING I

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

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

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

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

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

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

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.,

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

- 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.

- There are 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 APPROACH

- 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.
- The main elements of the aggregation approach:
  - Introduce a few "aggregate" states, viewed as the states of an "aggregate" system
  - Define transition probabilities and costs of the aggregate system, by associating multiple states of the original system with each aggregate state
  - Solve (exactly or approximately) 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 to obtain an approximation of the optimal cost of the original problem
- Example (Hard Aggregation): We are given a partition of the state space into subsets of states, and each subset is viewed as an aggregate state (each state belongs to one and only one subset).

GGREGATION/DISAGGREGATION PROBABILITIE

- The aggregate system transition probabilities are defined via two (somewhat arbitrary) choices:

- For each original system state i and aggregate state m, the aggregation probability a_{im} (we have $\sum_m a_{im} = 1$ for each i).
  - This may be roughly interpreted as the "degree of membership of i in the aggregate state m."
  - In the hard aggregation example, $a_{im} = 1$ if state i belongs to aggregate state/subset m.

- For each aggregate state m and original system state i, the disaggregation probability d_{mi} (we have $\sum_i d_{mi} = 1$ for each m).
  - This may be roughly interpreted as the "degree to which i is representative of m."
  - In the hard aggregation example (assuming all states that belong to aggregate state/subset m are "equally representative") $d_{mi} = 1/|m|$ for each state i that belongs to aggregate state/subset m, where $|m|$ is the cardinality (number of states) of m.

AGGREGATE TRANSITION PROBABILITIES

- Given the aggregation and disaggregation probabilities, a_{im} and d_{mi}, and the original transition probabilities $p_{ij}(u)$, the transition probability from aggregate state m to aggregate state n under u, and corresponding transition cost, are given by:

$$q_{mn}(u) = \sum_i \sum_j d_{mi} p_{ij}(u) a_{jn}$$

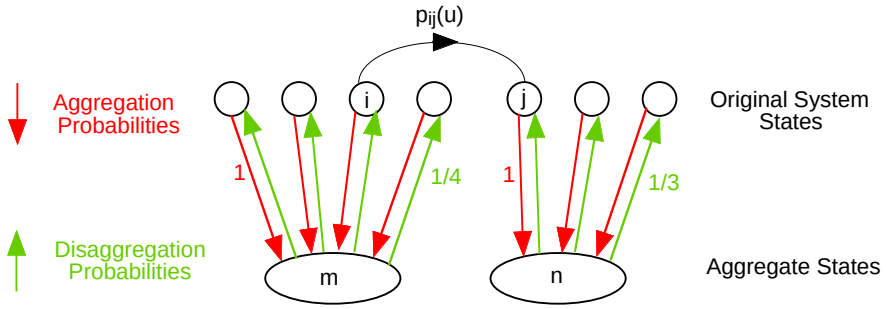
- This corresponds to a probabilistic process that can be simulated as follows:
  - From aggregate state m, generate original state i according to d_{mi}.
  - Generate a transition from i to j according to $p_{ij}(u)$, with cost $g(i, u, j)$.
  - From original state j, generate aggregate state n according to a_{jn}.

- After solving for the optimal costs $\hat{J}(m)$ of the aggregate problem, the costs of the original problem are approximated by

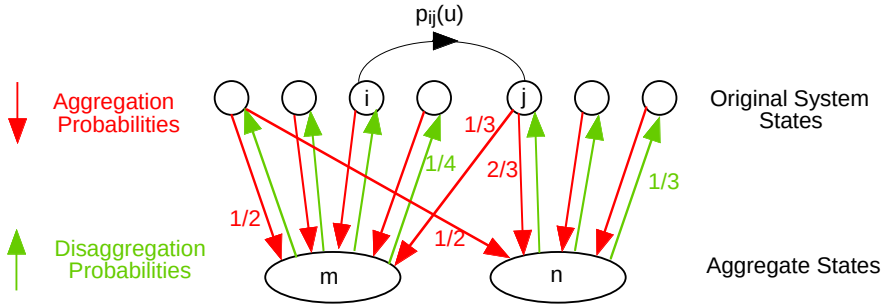
$$\tilde{J}(i) = \sum_m a_{im} \hat{J}(m)$$

AGGREGATION EXAMPLES

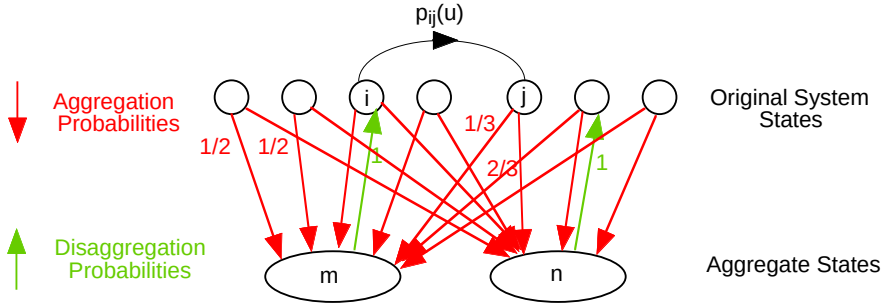
- Hard aggregation (each original system state is associated with one aggregate state):



- Soft aggregation (each original system state is associated with multiple aggregate states):



- Coarse grid (each aggregate state is an original system state):



APPROXIMATE LINEAR PROGRAMMING

- Approximate J^* using a linear architecture

$$\tilde{J} = \Phi r$$

where $r = (r_1, \ldots, r_s)$ is a weight vector, and Φ is an $n \times s$ feature matrix.

- Use $\tilde{J}$ in place of J^* in the linear programming approach, i.e., compute r by solving

$$\begin{aligned} &\text{maximize} \quad c'\Phi r \\ &\text{subject to} \quad \Phi r \leq g_\mu + \alpha P_\mu \Phi r, \quad \forall\ \mu \end{aligned}$$

where c is a vector with positive components.

- This is a linear program with s variables but an enormous number of constraints (one constraint for each state-control pair).

- Special large-scale linear programming methods (cutting plane or column generation methods) may be used for such problems.

- Approximations using only a "sampled" subset of state-control pairs are possible (see the papers by de Farias and Van Roy).

APPROXIMATION IN POLICY SPACE I

- Consider an average cost problem, where the problem data are parameterized by a vector r, i.e., a cost vector $g(r)$, transition probability matrix $P(r)$. Let $\lambda(r)$ be the (scalar) average cost per stage, satisfying Bellman's equation

$$\lambda(r)e + v(r) = g(r) + P(r)v(r)$$

- Consider minimizing $\lambda(r)$ over r (here the data dependence on control is encoded in the parameterization). We can try to solve the problem by nonlinear programming/gradient descent methods.

- **Important fact:** If $\Delta\lambda$ is the change in λ due to a small change Δr from a given r, we have

$$\Delta\lambda \cdot e = \overline{p}'(\Delta g + \Delta P v),$$

where $\overline{p}$ is the steady-state probability distribution/vector corresponding to $P(r)$, and all the quantities above are evaluated at r:

$$\Delta\lambda = \lambda(r + \Delta r) - \lambda(r),$$

$$\Delta g = g(r+\Delta r) - g(r), \qquad \Delta P = P(r+\Delta r) - P(r)$$

APPROXIMATION IN POLICY SPACE II

- **Proof of the gradient formula:** We have, by "differentiating" Bellman's equation,

$$\Delta\lambda(r)\cdot e+\Delta v(r) = \Delta g(r)+\Delta P(r)v(r)+P(r)\Delta v(r)$$

By left-multiplying with $\overline{p}'$,

$$\overline{p}'\Delta\lambda(r)\cdot e+\overline{p}'\Delta v(r) = \overline{p}'\Big(\Delta g(r)+\Delta P(r)v(r)\Big)+\overline{p}'P(r)\Delta v(r)$$

Since $\overline{p}'\Delta\lambda(r)\cdot e = \Delta\lambda(r)e$ and $\overline{p}' = \overline{p}'P(r)$, this equation simplifies to

$$\Delta\lambda\cdot e = \overline{p}'(\Delta g + \Delta P v)$$

- Since we don't know $\overline{p}$, we cannot implement a gradient-like method for minimizing $\lambda(r)$. An alternative is to use "sampled gradients", i.e., generate a simulation trajectory $(i_0, i_1, \ldots)$, and change r once in a while, in the direction of a simulation-based estimate of $\overline{p}'(\Delta g + \Delta P v)$.

- There is much recent research on this subject, see e.g., the work of Marbach and Tsitsiklis, and Konda and Tsitsiklis, and the refs given there.