LECTURE SLIDES ON DYNAMIC PROGRAMMIN

BASED ON LECTURES GIVEN AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOG

CAMBRIDGE, MASS

FALL 2009

DIMITRI P. BERTSEKAS

These lecture slides are based on the book: "Dynamic Programming and Optimal Control: 3rd edition," Vols. 1 and 2, Athena Scientific, 2007, by Dimitri P. Bertsekas; see

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

Last Updated: September 2009

6.231 DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

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

DP AS AN OPTIMIZATION METHODOLOGY

- Generic optimization problem:

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

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

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

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

    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

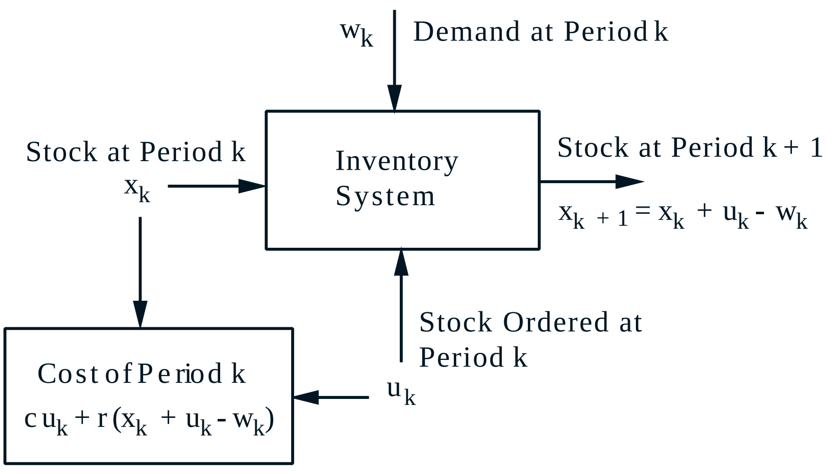
$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1$$

  - 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

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

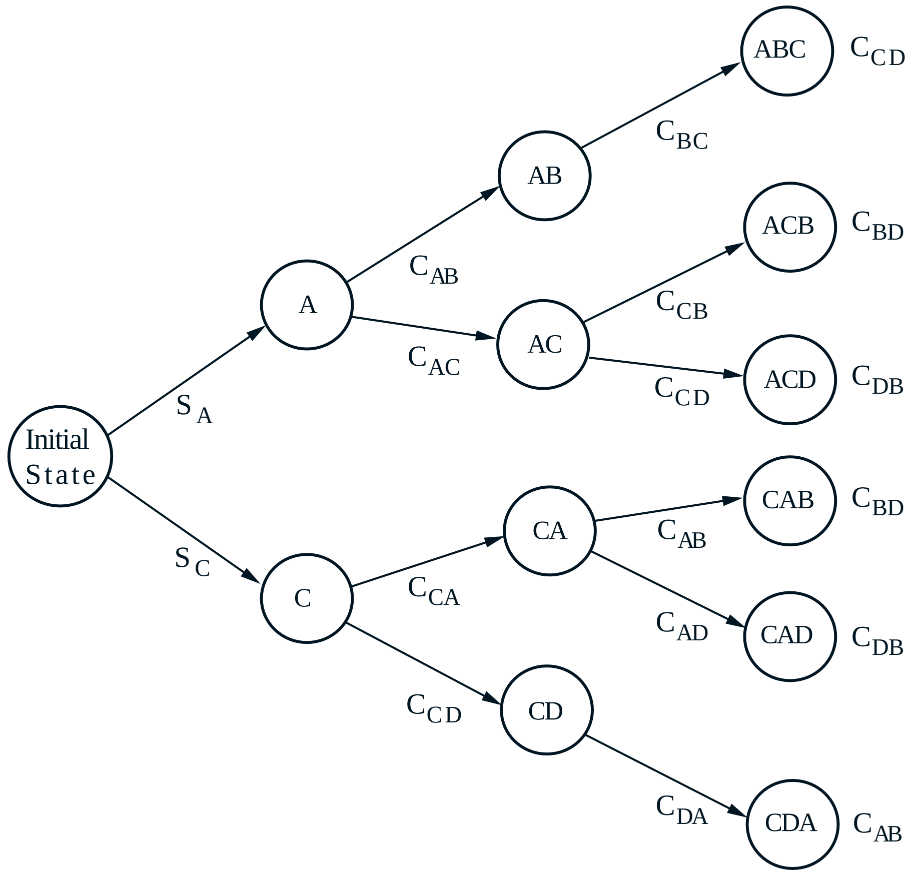
- Optimization over policies: Rules/functions $u_k = \mu_k(x_k)$ that map states to controls

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.,
  $$u_k \in U_k(x_k)$$
  - w_k is random and generated according to a distribution
  $$P_{w_k}(x_k, u_k)$$

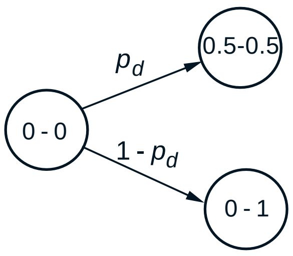
DETERMINISTIC FINITE-STATE PROBLEMS

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

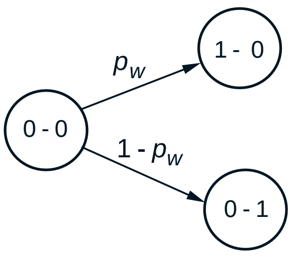


STOCHASTIC FINITE-STATE PROBLEMS

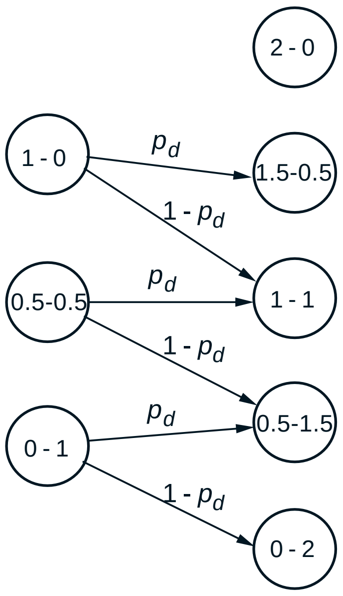
- Example: Find two-game chess match strategy
- *Timid* play draws with prob. $p_d > 0$ and loses with prob. $1-p_d$. *Bold* play wins with prob. $p_w < 1/2$ and loses with prob. $1-p_w$



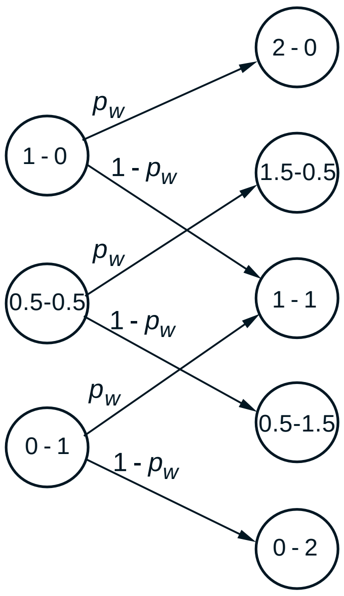
1st Game / Timid Play



1st Game / Bold Play



2nd Game / Timid Play



2nd Game / Bold Play

BASIC PROBLEM

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

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

- Optimal cost function

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

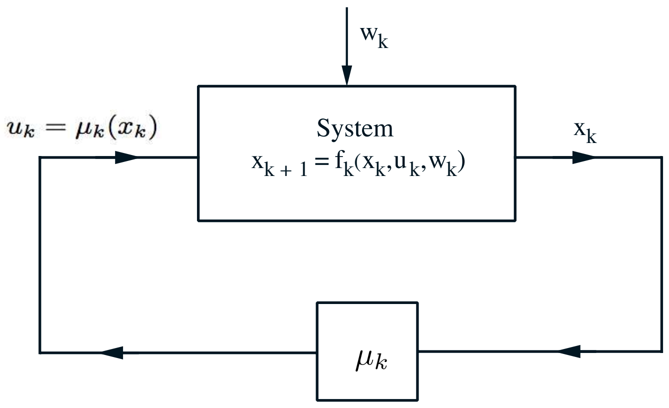
- Optimal policy π^* satisfies

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

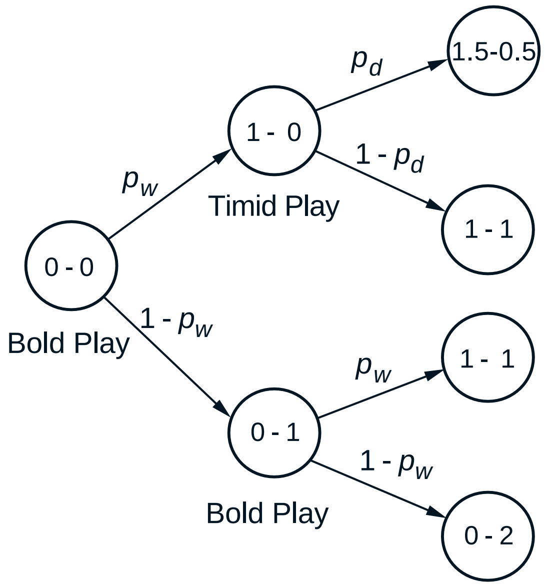
When produced by DP, π^* is independent of x_0.

SIGNIFICANCE OF FEEDBACK

- Open-loop versus closed-loop policies



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



VARIANTS OF DP PROBLEMS

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

LECTURE BREAKDOWN

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

A NOTE ON THESE SLIDES

- These slides are a teaching aid, not a text
- Don't expect a rigorous mathematical development or precise mathematical statements
- Figures are meant to convey and enhance ideas, not to express them precisely
- Omitted proofs and a much fuller discussion can be found in the 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 $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \dots, N-1$
- Control constraints $u_k \in U_k(x_k)$
- Probability distribution $P_k(\cdot \mid x_k, u_k)$ of w_k
- Policies $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k
- Expected cost of π starting at x_0 is

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

- Optimal cost function

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

- Optimal policy π^* is one that satisfies

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

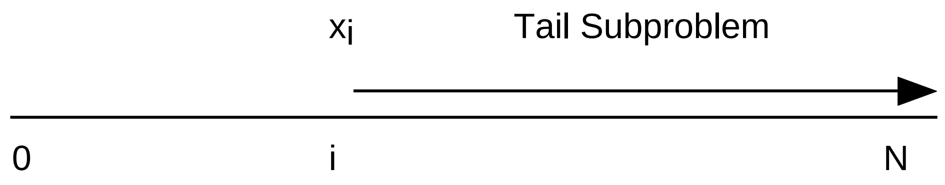
PRINCIPLE OF OPTIMALITY

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

- Consider the "tail subproblem" whereby we are at x_i at time i and wish to minimize the "cost-to-go" from time i to time N

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

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



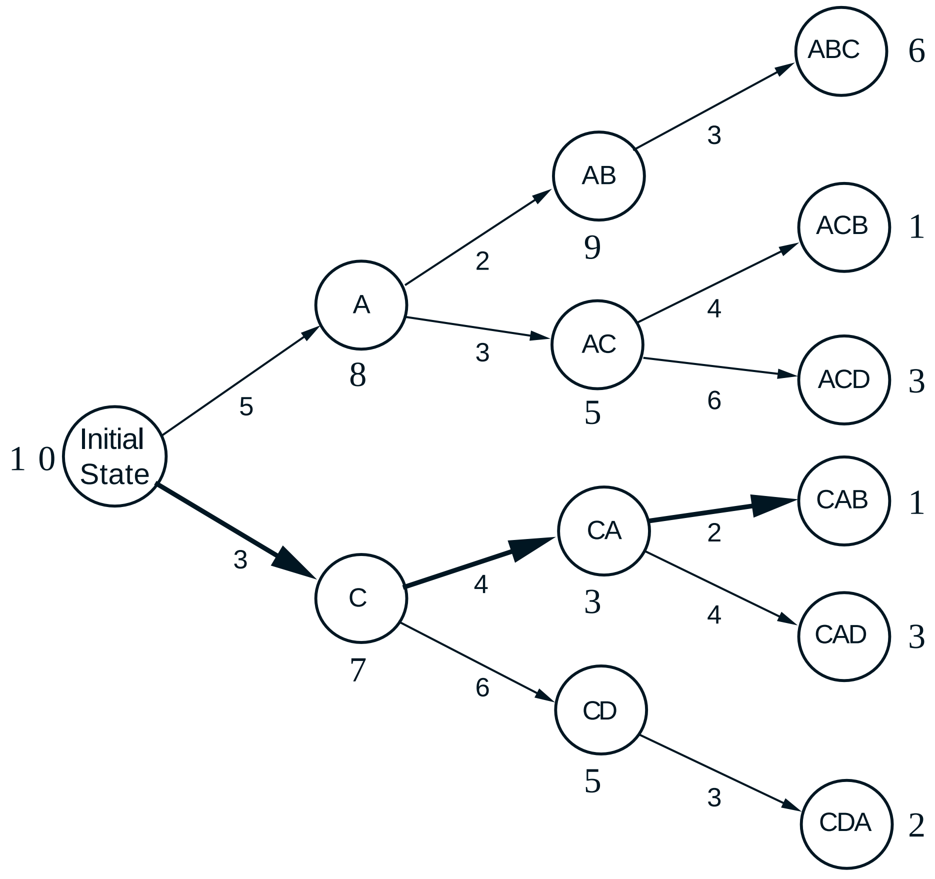
- *Principle of optimality*: The tail policy is optimal for the tail subproblem (optimization of the future does not depend on what we did in the past)

- 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

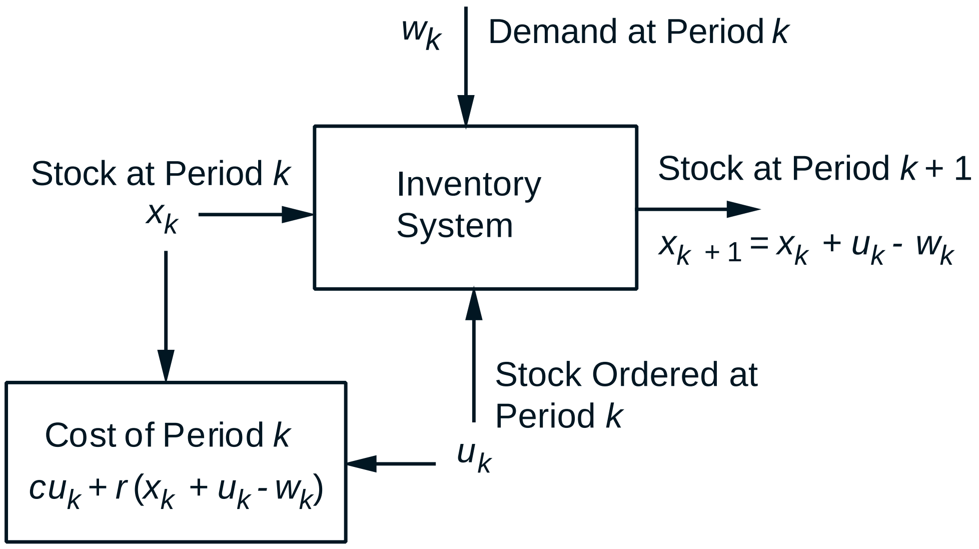
DETERMINISTIC SCHEDULING EXAMPLE

- 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:

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

- Tail Subproblems of Length $N-k$:

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

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

DP ALGORITHM

- Start with

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

and go backwards using

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

- 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:
  - ALL the tail subproblems are solved (in addition to the original problem)
  - Intensive computational requirements

PROOF OF THE INDUCTION STEP

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

$$
\begin{aligned}
J^*_k(x_k) = \min_{(\mu_k,\pi_{k+1})} \mathop{E}_{w_k,\ldots,w_{N-1}} \Bigg\{ & g_k\Big(x_k,\mu_k(x_k),w_k\Big) \\
& + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\Big(x_i,\mu_i(x_i),w_i\Big) \Bigg\}
\end{aligned}
$$

$$
= \min_{\mu_k} \mathop{E}_{w_k} \Bigg\{ g_k\Big(x_k,\mu_k(x_k),w_k\Big)
$$

$$
+ \min_{\pi_{k+1}} \Bigg[\mathop{E}_{w_{k+1},\ldots,w_{N-1}} \Bigg\{ g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\Big(x_i,\mu_i(x_i),w_i\Big) \Bigg\} \Bigg]
$$

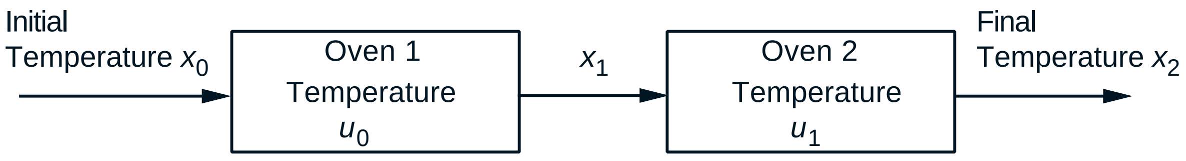
$$
= \min_{\mu_k} \mathop{E}_{w_k} \Big\{ g_k\Big(x_k,\mu_k(x_k),w_k\Big) + J^*_{k+1}\Big(f_k\Big(x_k,\mu_k(x_k),w_k\Big)\Big)\Big\}
$$

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

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

$$
= J_k(x_k)
$$

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:

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

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

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

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:

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

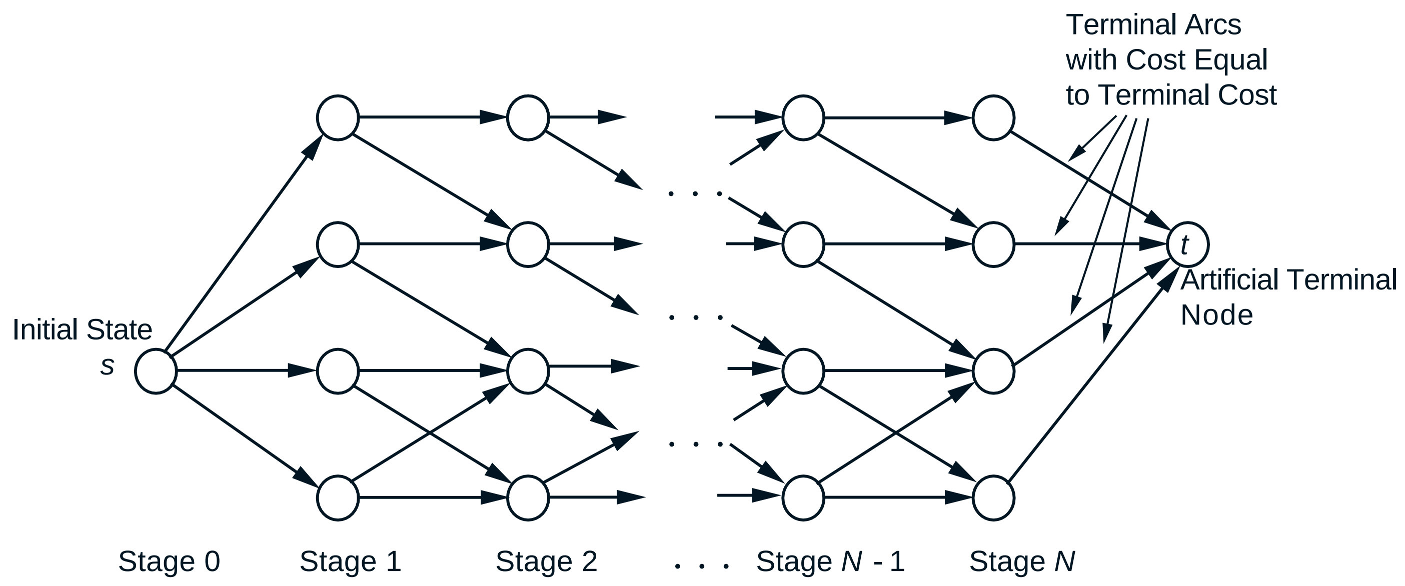
6.231 DYNAMIC PROGRAMMING

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:

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

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

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):

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

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

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

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

A NOTE ON FORWARD DP ALGORITHMS

- There is no forward DP algorithm for stochastic problems

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

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

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

GENERIC SHORTEST PATH PROBLEMS

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

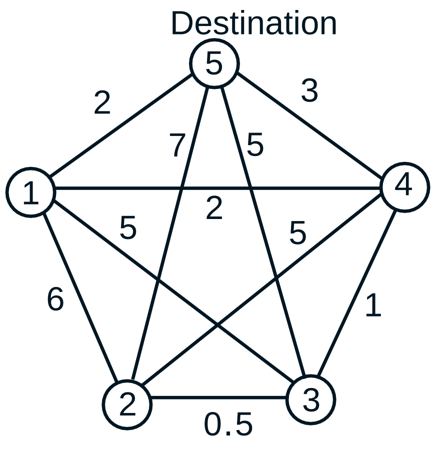
$J_0(i)$: Cost of the optimal path from i to t.

- DP algorithm:

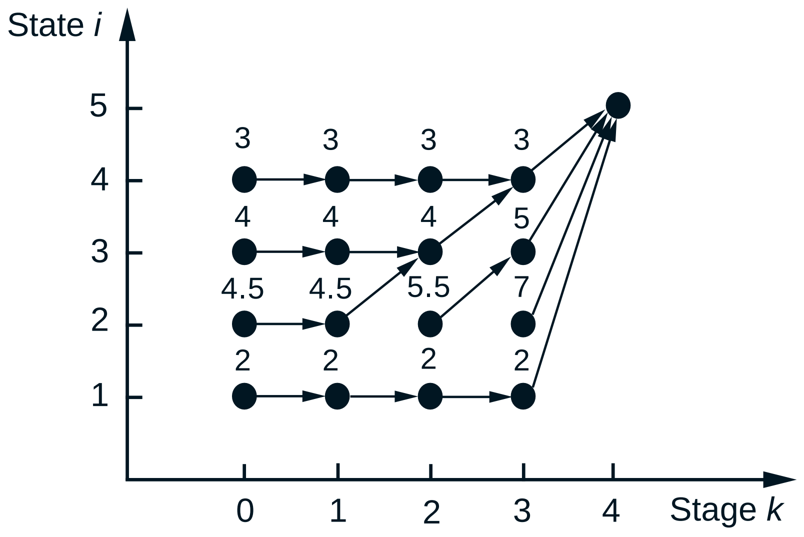
$$J_k(i) = \min_{j=1,\ldots,N} \big[a_{ij} + J_{k+1}(j)\big], \qquad k = 0, 1, \ldots, N-2,$$

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

EXAMPLE



(a)



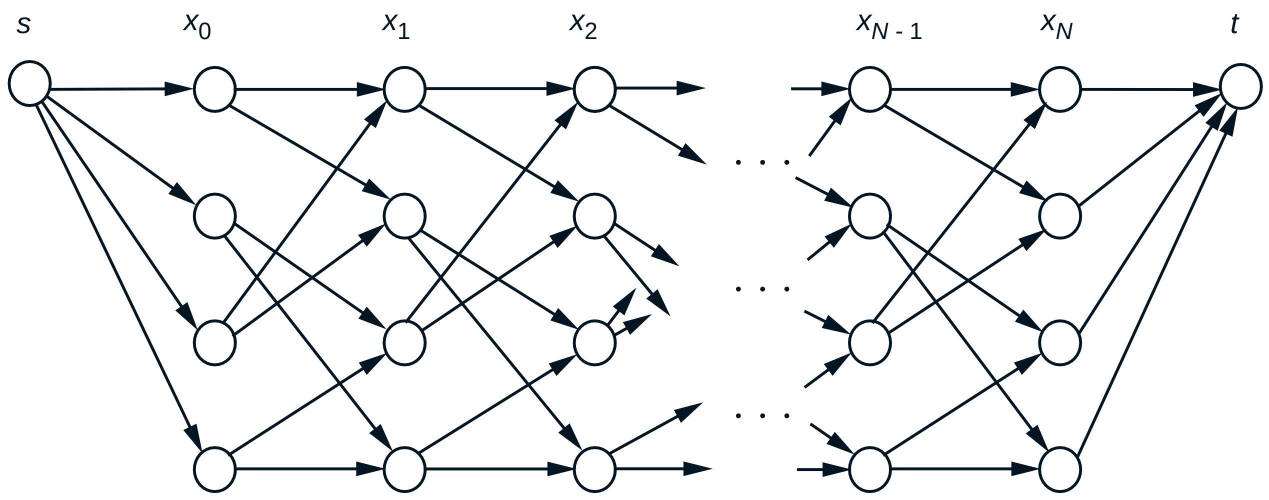
(b)

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

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

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

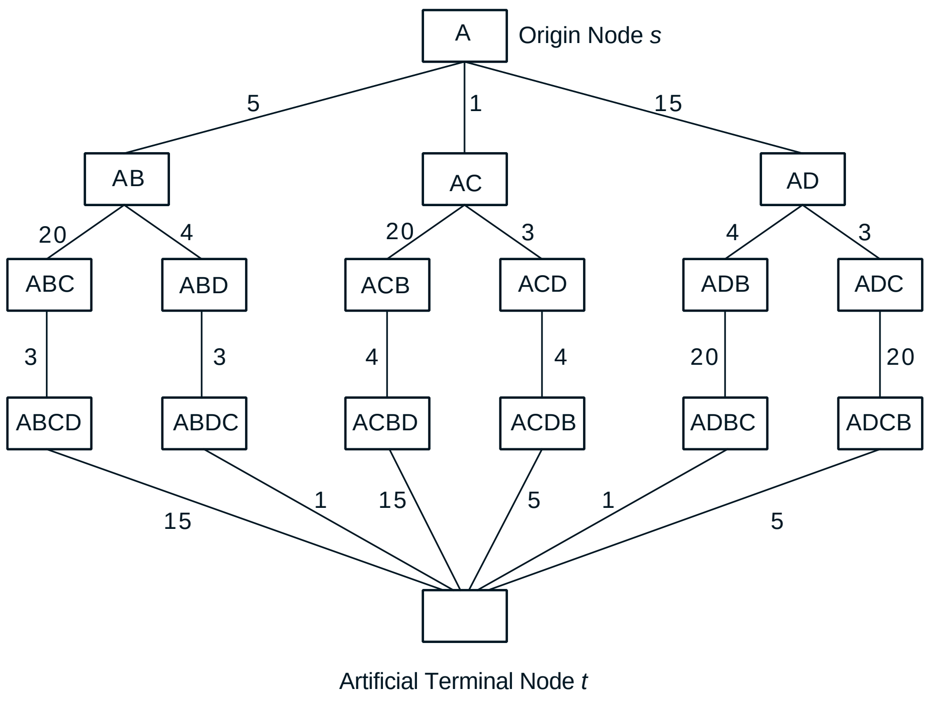
- Maximizing $p(X_N \mid Z_N)$ is equivalent with maximizing $\ln(p(X_N, Z_N))$

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

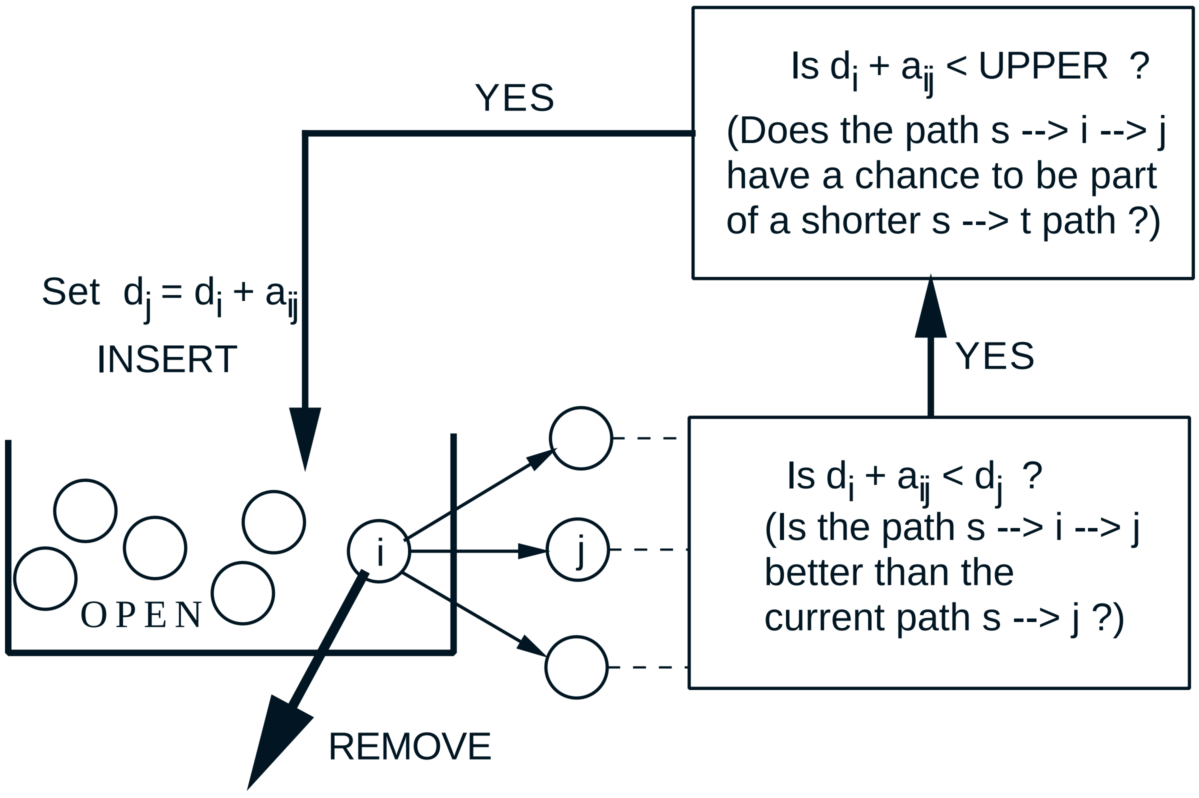
Step 1 (Node Removal): Remove a node i from OPEN and for each child j of i, do step 2

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

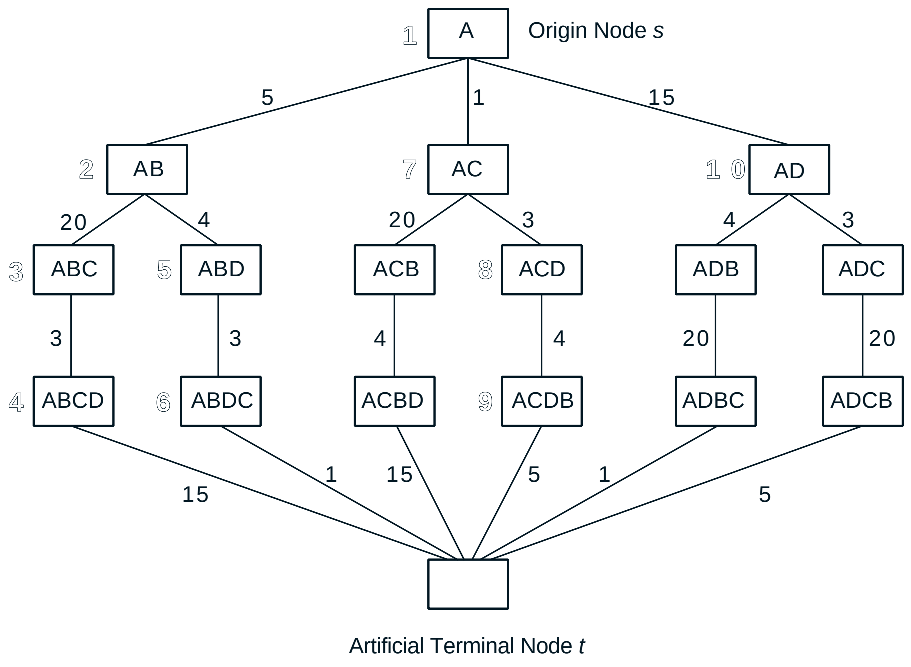
Step 3 (Termination Test): If OPEN is empty, terminate; else go to step 1

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


Is $d_i + a_{ij}$ < UPPER ?
(Does the path s --> i --> j have a chance to be part of a shorter s --> t path ?)
YES
YES
Set $d_j = d_i + a_{ij}$
INSERT
OPEN
i
j
REMOVE
Is $d_i + a_{ij} < d_j$?
(Is the path s --> i --> j better than the current path s --> j ?)

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

6.231 DYNAMIC PROGRAMMING

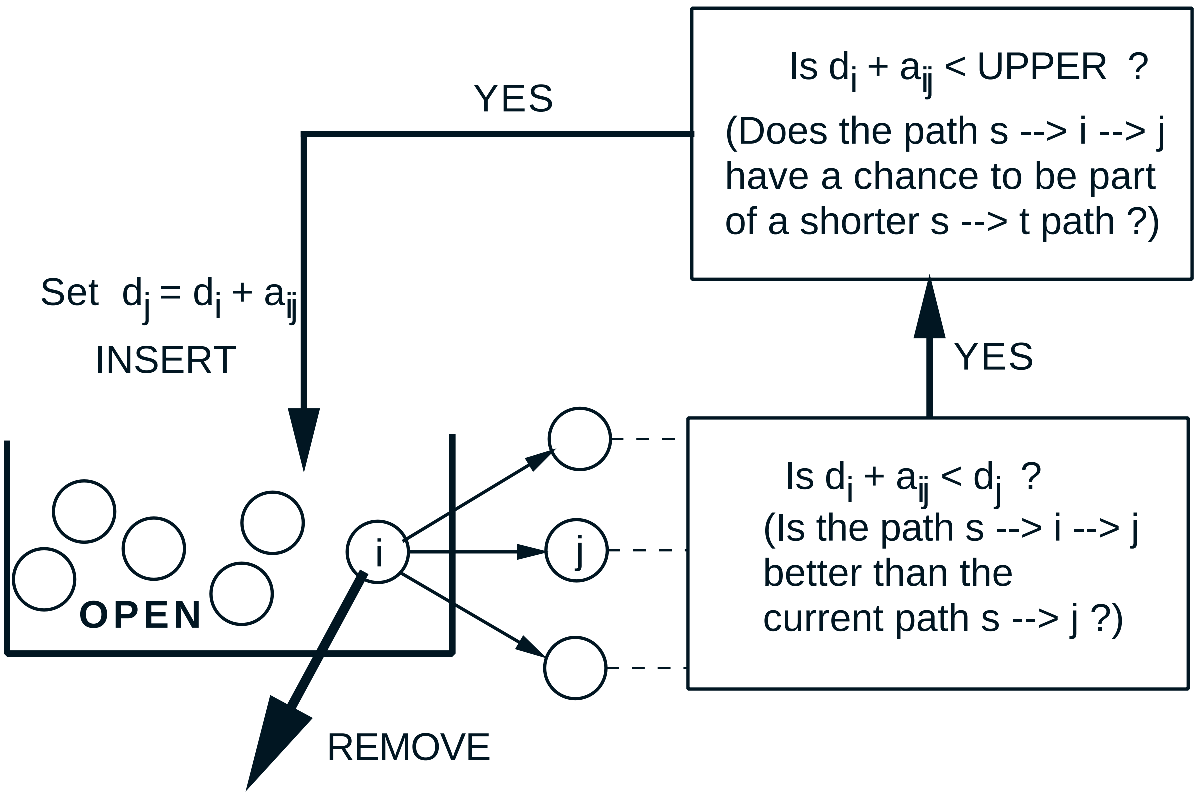
LECTURE 4

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=$\{s\}$)


Is $d_i + a_{ij}$ < UPPER ?
(Does the path s --> i --> j have a chance to be part of a shorter s --> t path ?)
YES
Set $d_j = d_i + a_{ij}$
INSERT
YES
OPEN
i
j
Is $d_i + a_{ij} < d_j$?
(Is the path s --> i --> j better than the current path s --> j ?)
REMOVE

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

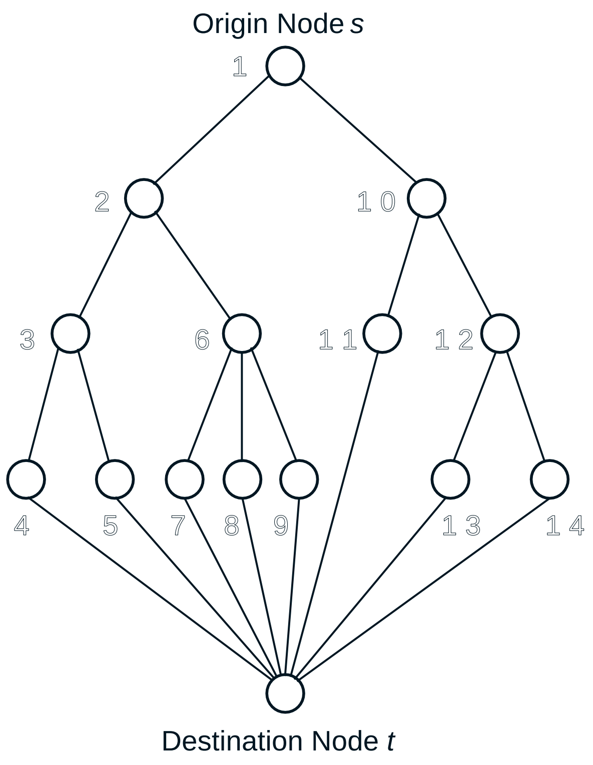
(3) Let $(s, j_1, j_2, \ldots, j_k, t)$ be a shortest path and let d^* be the shortest distance. If UPPER $> d^*$ at termination, UPPER will also be larger than the length of all the paths $(s, j_1, \ldots, j_m)$, $m = 1, \ldots, k$, throughout the algorithm. Hence, node j_k will never enter the OPEN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

ADVANCED INITIALIZATION

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

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

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

- 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

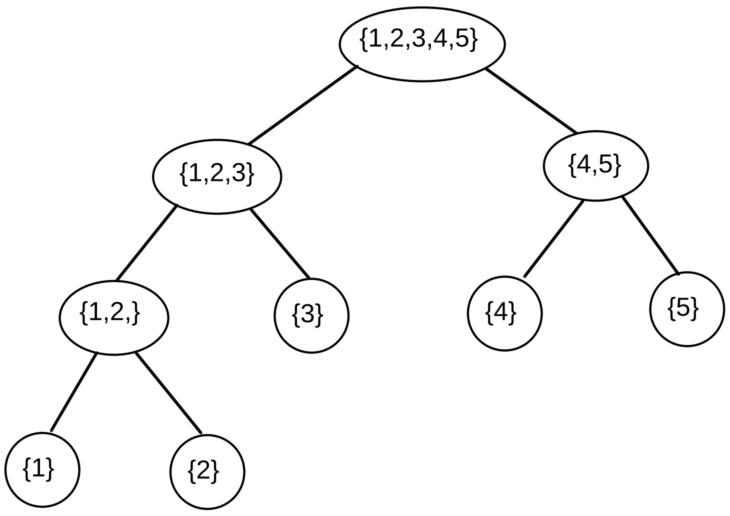
$$\underline{f}_1 \leq \min_{x \in Y_1} f(x), \qquad \overline{f}_2 \geq \min_{x \in Y_2} f(x).$$

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 $f(x) - \underline{f}_X$
- Shortest path from X to the set of leafs gives the optimal cost and optimal solution
- UPPER is the smallest $f(x) - \underline{f}_X$ 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:

- Entry Test: If $\underline{f}_{Y_j} <$ UPPER, place Y_j in OPEN.
- Update UPPER: If $\overline{f}_{Y_j} <$ UPPER, set UPPER $= \overline{f}_{Y_j}$, and if Y_j consists of a single solution, mark that as being the best solution found so far

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

- 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 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

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

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

$$K_N = Q_N,$$

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

- 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 EQUATIC

- 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

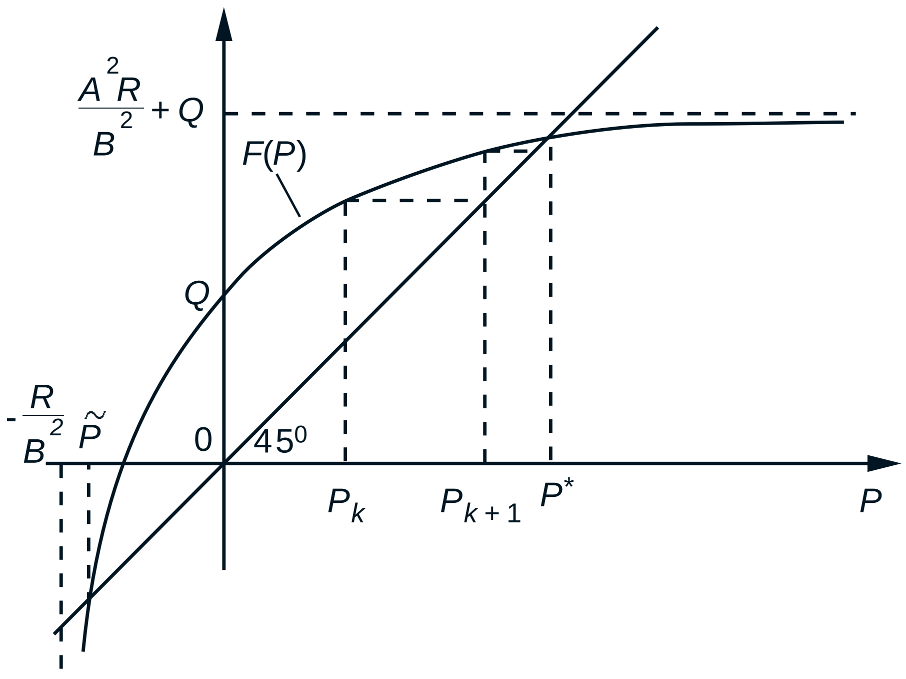
$$L = -(B'KB + R)^{-1}B'KA,$$

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 $P_k = K_{N-k}$):

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

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

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

- 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

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

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

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

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

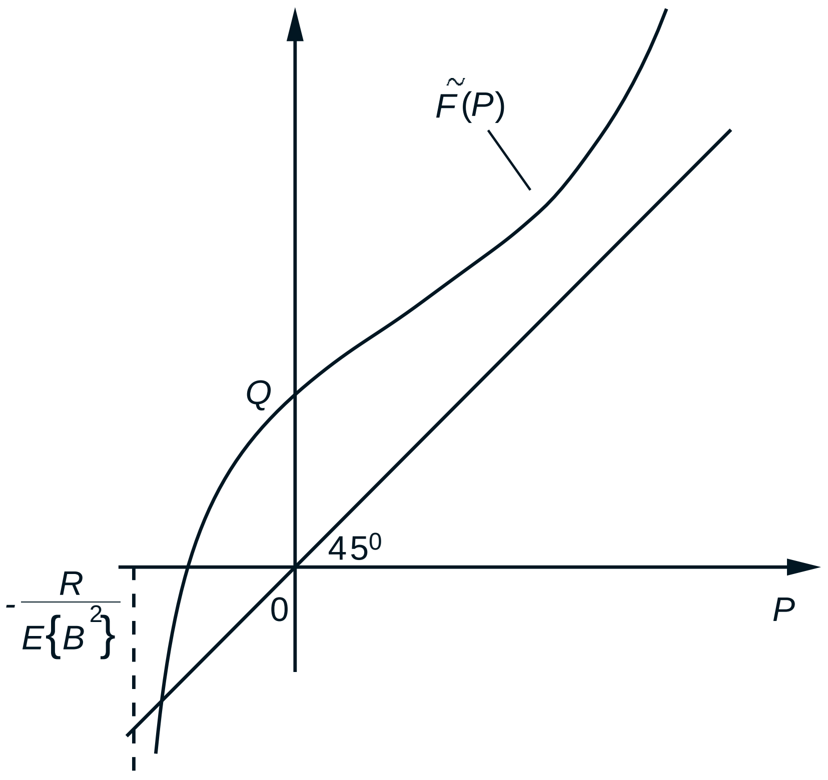
and where the matrices K_k are given by

$$K_N = Q_N,$$

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

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

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

- 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

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

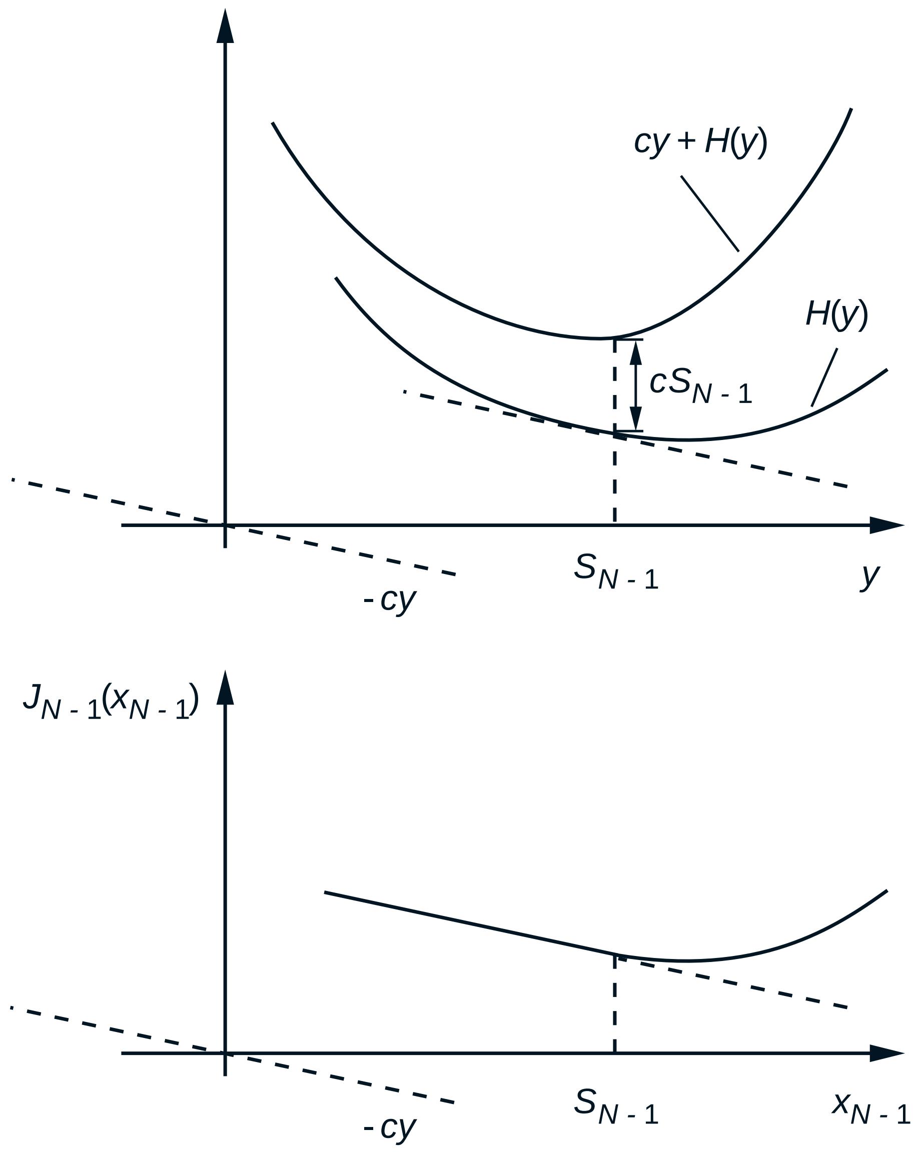
where S_k minimizes $G_k(y)$.

- 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

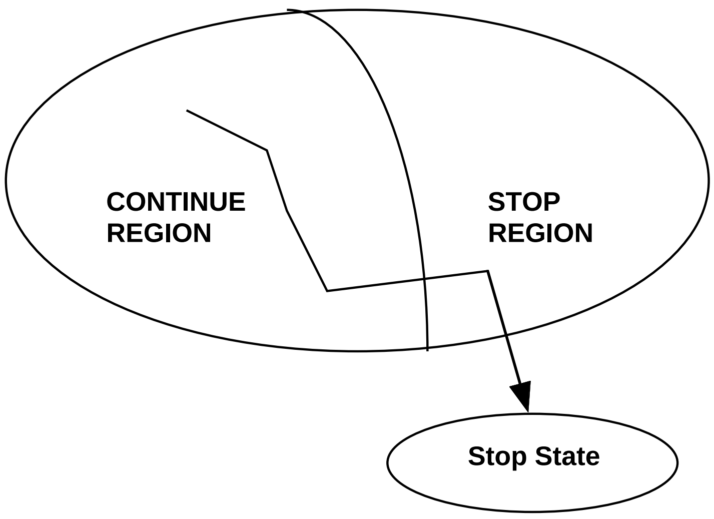
LECTURE 6

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;

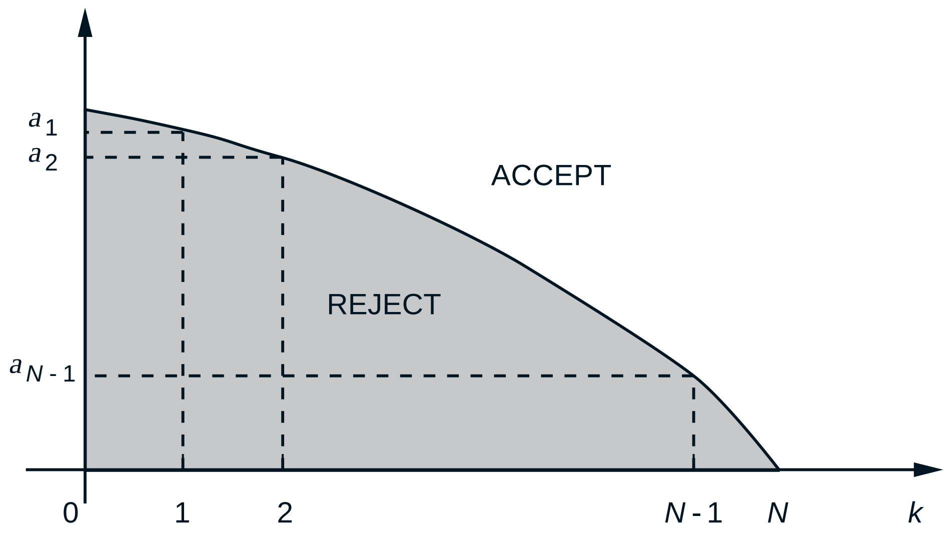
$$\text{accept the offer } x_k \qquad \text{if } x_k > \alpha_k,$$

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

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

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

We have $\alpha_k = E_w\big\{V_{k+1}(w)\big\}/(1+r)$, 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

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

- Since $J_{N-1}(x) \le J_N(x)$, we have $J_k(x) \le J_{k+1}(x)$ 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

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

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

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

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

BASIC PROBLEM WITH IMPERFECT STATE INF

- 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

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

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

REFORMULATION AS PERFECT INFO PROBLEM

- We have

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

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

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

so the cost per stage of the new system is

$$\tilde{g}_k(I_k, u_k) = \mathop{E}_{x_k, w_k} \big\{g_k(x_k, u_k, w_k) \mid I_k, u_k\big\}$$

DP ALGORITHM

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

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

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

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

- 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

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

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

- Observations

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

- $w_0, \ldots, w_{N-1}$, $v_0, \ldots, v_{N-1}$ 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 $N-1$):

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

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

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

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

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

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

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

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

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

DP ALGORITHM III

- DP equation for period $N-2$:

$$\begin{aligned} J_{N-2}(I_{N-2}) = \min_{u_{N-2}} \Big[& \mathop{E}_{x_{N-2},w_{N-2},z_{N-1}} \{x'_{N-2}Qx_{N-2} \\ & + u'_{N-2}Ru_{N-2} + J_{N-1}(I_{N-1}) \mid I_{N-2}, u_{N-2}\} \Big] \\ = E\Big\{ & x'_{N-2}Qx_{N-2} \mid I_{N-2}\Big\} \\ & + \min_{u_{N-2}} \Big[u'_{N-2}Ru_{N-2} \\ & \qquad + E\Big\{x'_{N-1}K_{N-1}x_{N-1} \mid I_{N-2}, u_{N-2}\Big\}\Big] \\ & + E\Big\{ \Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big)' \\ & \quad \cdot P_{N-1}\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big) \mid I_{N-2}, u_{N-2}\Big\} \\ & + E_{w_{N-1}}\{w'_{N-1}Q_N w_{N-1}\} \end{aligned}$$

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

QUALITY OF ESTIMATION LEMMA

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

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

independently of the policy being used

- 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)

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

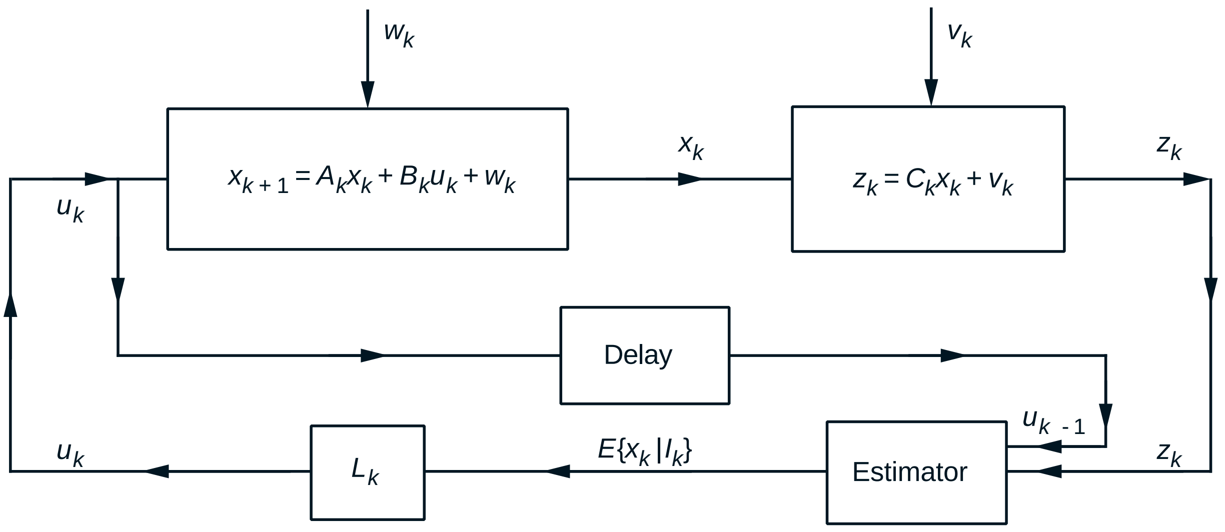
where L_k is the same as for perfect state info:

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

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 8

LECTURE OUTLINE

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

REVIEW: IMPERFECT STATE INFO PROBLEM

- Instead of knowing x_k, we receive observations

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

- I_k: information vector available at time k:

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

- 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

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

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 $k = N-1$,

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

- 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

$$S_k(I_k) = P_{x_k \mid I_k}$$

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

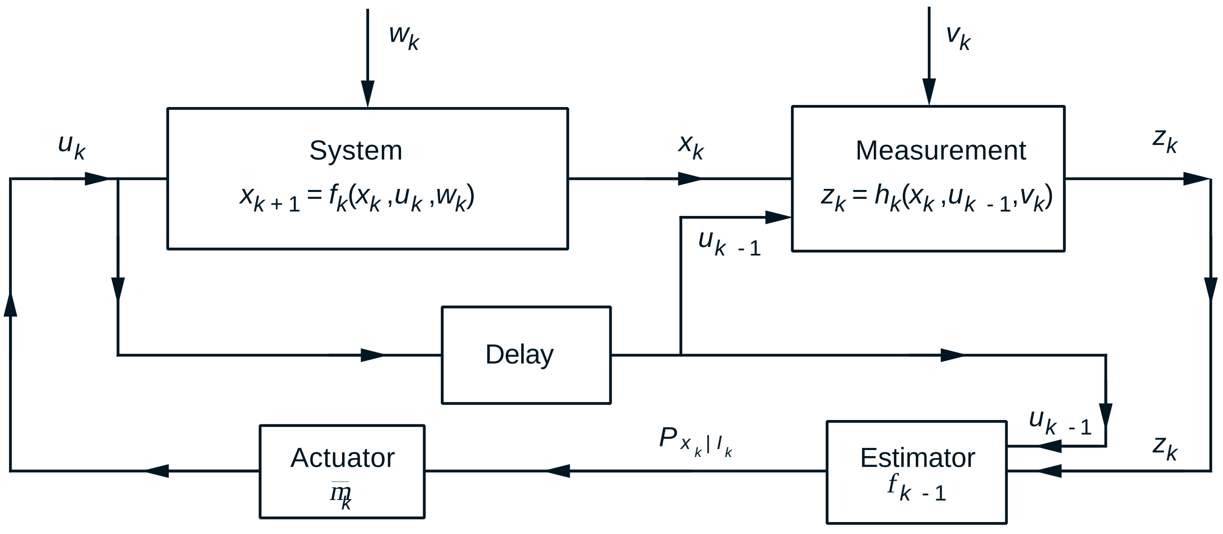
- It turns out that $P_{x_k|I_k}$ is generated recursively by a dynamic system (estimator) of the form

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

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

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

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

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

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

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

$$p_k\beta V \ge C$$

(expected reward from the next search $\ge$ the cost of the search)

FINITE-STATE SYSTEMS - POMDP

- Suppose the system is a finite-state Markov chain, with states $1, \ldots, n$.

- Then the conditional probability distribution $P_{x_k \mid I_k}$ 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 the problem is called a POMDP (Partially Observed Markov Decision Problem).

- For POMDP 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. It is also useful in practice because it forms the basis for approximations.

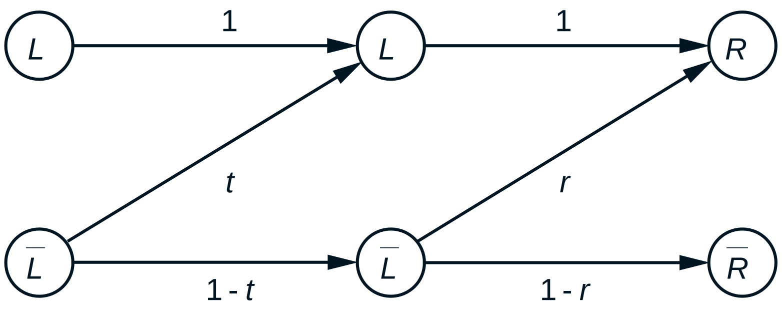
INSTRUCTION EXAMPLE

- 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:

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

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

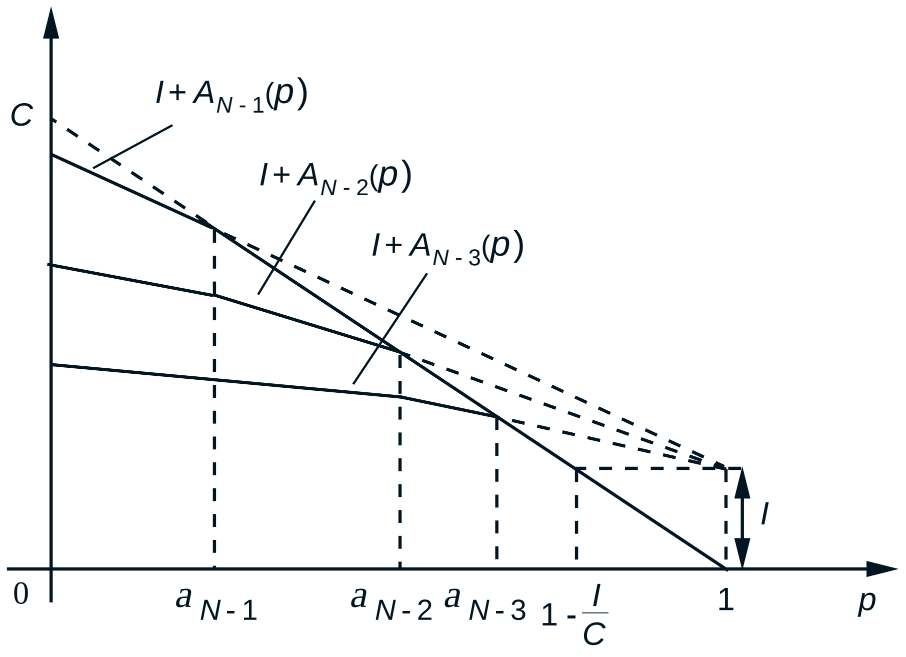
$$\overline{J}_k(p_k) = \min\big[(1-p_k)C,\, I + A_k(p_k)\big],$$

where

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

- 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 9

LECTURE OUTLINE

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

PRACTICAL DIFFICULTIES OF DP

- The curse of 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

COST-TO-GO FUNCTION APPROXIMATION

- Use a policy computed from the DP equation where the optimal cost-to-go function J_{k+1} is replaced by an approximation $\tilde{J}_{k+1}$. (Sometimes $E\{g_k\}$ is also replaced by an approximation.)
- Apply $\overline{\mu}_k(x_k)$, which attains the minimum in

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

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

CERTAINTY EQUIVALENT CONTROL (CEC)

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

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

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

  where x_k is known, and

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

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

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

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

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

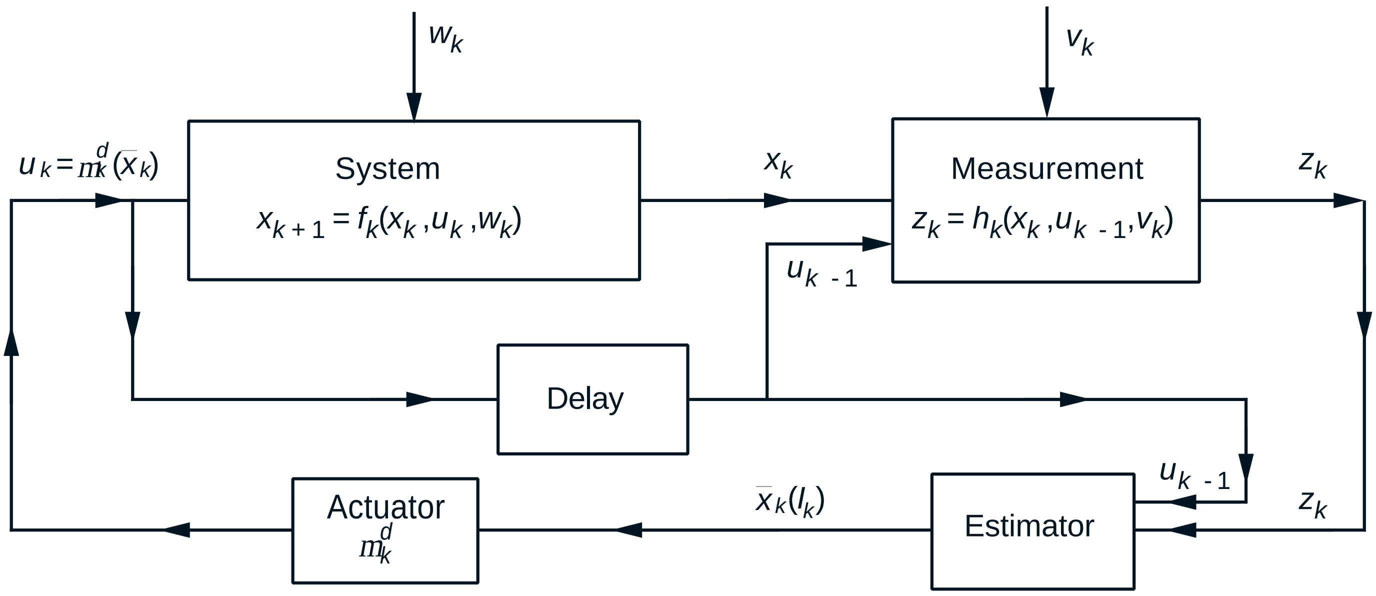
ALTERNATIVE OFF-LINE IMPLEMENTATION

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

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

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

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



PARTIALLY STOCHASTIC CEC

- Instead of fixing *all* future disturbances to their typical values, fix only some, and treat the rest as stochastic.

- **Important special case:** Treat an imperfect state information problem as one of perfect state information, using an estimate $\overline{x}_k(I_k)$ 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).

GENERAL COST-TO-GO APPROXIMATION

- **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 FOR 1SL

- 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

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

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

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

- **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

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

COMPUTATIONAL ASPECTS

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

PROBLEM APPROXIMATION

- Many (problem-dependent) possibilities
  - Replace uncertain quantities by nominal values, or simplify the calculation of expected values by limited simulation
  - Simplify difficult constraints or dynamics

- Example of enforced decomposition: Route m vehicles that move over a graph. Each node has a "value." The first vehicle that passes through the node collects its value. Max the total collected value, subject to initial and final time constraints (plus time windows and other constraints).

- Usually the 1-vehicle version of the problem is much simpler. This motivates an approximation obtained by solving single vehicle problems.

- 1SL scheme: At time k and state x_k (position of vehicles and "collected value nodes"), consider all possible 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

PARAMETRIC COST-TO-GO APPROXIMATION

- Use a cost-to-go approximation from a parametric class $\tilde{J}(x, r)$ where x is the current state and $r = (r_1, \ldots, r_m)$ 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.

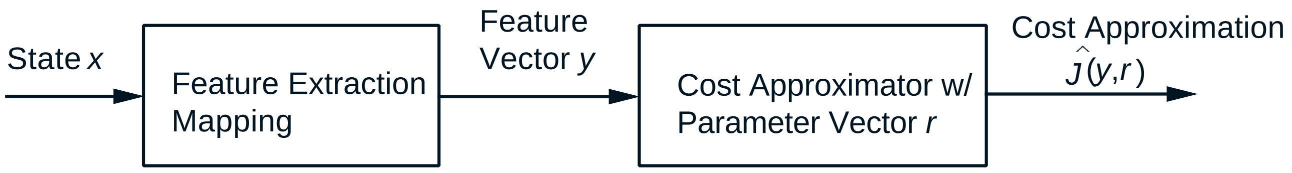
- Two key issues:
  - The choice of parametric class $\tilde{J}(x, r)$ (the approximation architecture).
  - Method for tuning the weights ("training" the architecture).

- 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

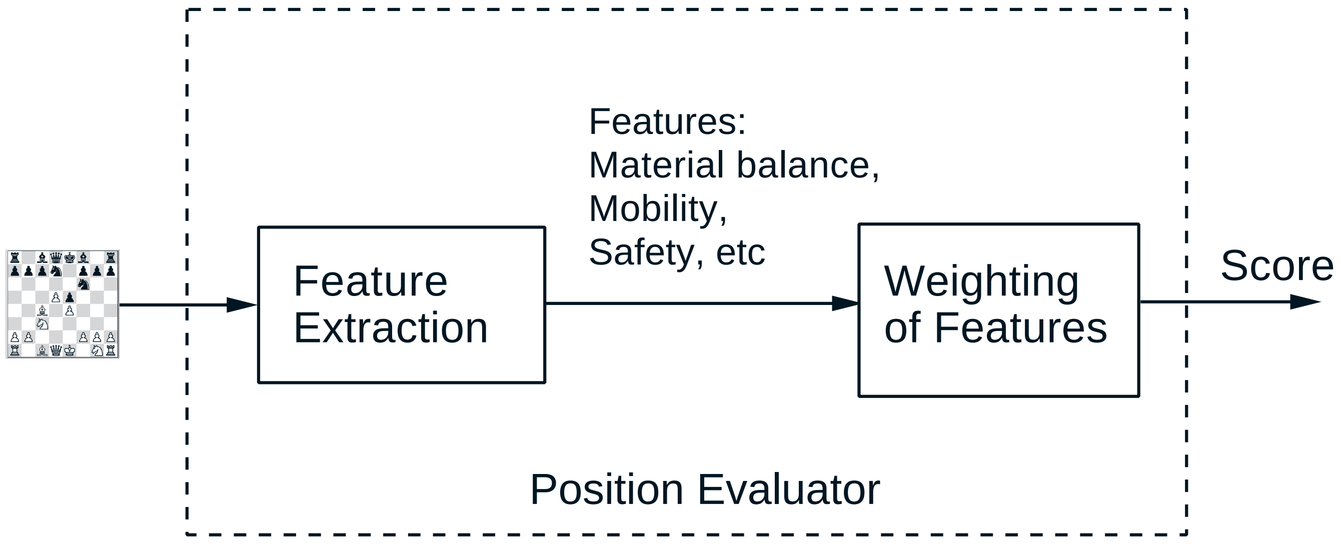

State x
Feature Extraction Mapping
Feature Vector y
Cost Approximator w/ Parameter Vector r
Cost Approximation $\hat J(y,r)$

- 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

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

AN EXAMPLE - COMPUTER CHESS

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



• Many context-dependent special features.

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

• Most often the training is done by trial and error.

6.231 DYNAMIC PROGRAMMING

LECTURE 10

LECTURE OUTLINE

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

COST IMPROVEMENT PROPERTY

- Let

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

$$H_k(x_k)\text{: Cost-to-go of the base policy}$$

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

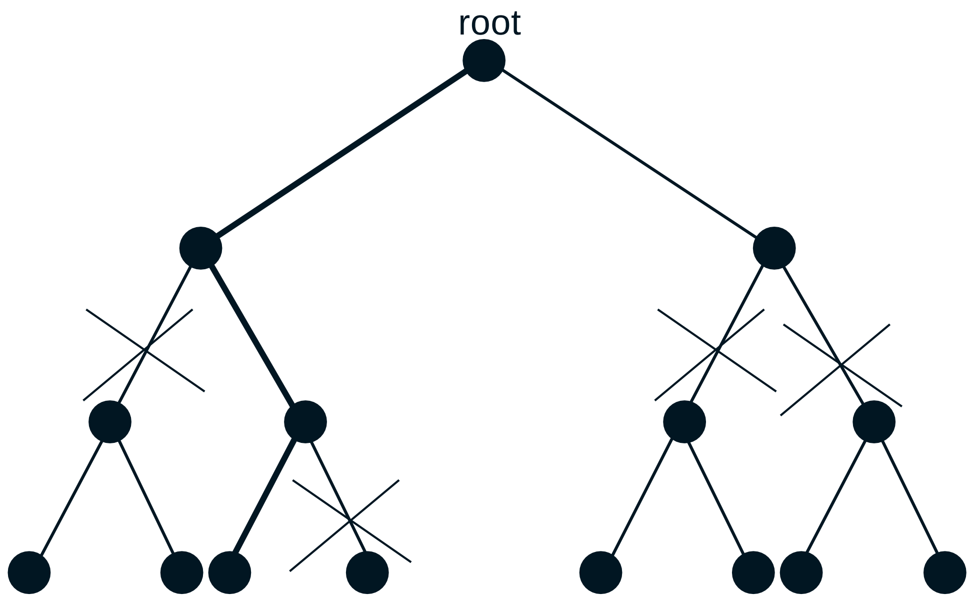
$$\overline{J}_{k+1}(x_{k+1}) \le H_{k+1}(x_{k+1}), \quad \forall\ x_{k+1}.$$

Then, for all x_k

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

- Induction hypothesis ==> 1st inequality
- Min selection of $\overline{\mu}_k(x_k)$ ==> 2nd inequality
- Definition of H_k, μ_k ==> last equality

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.

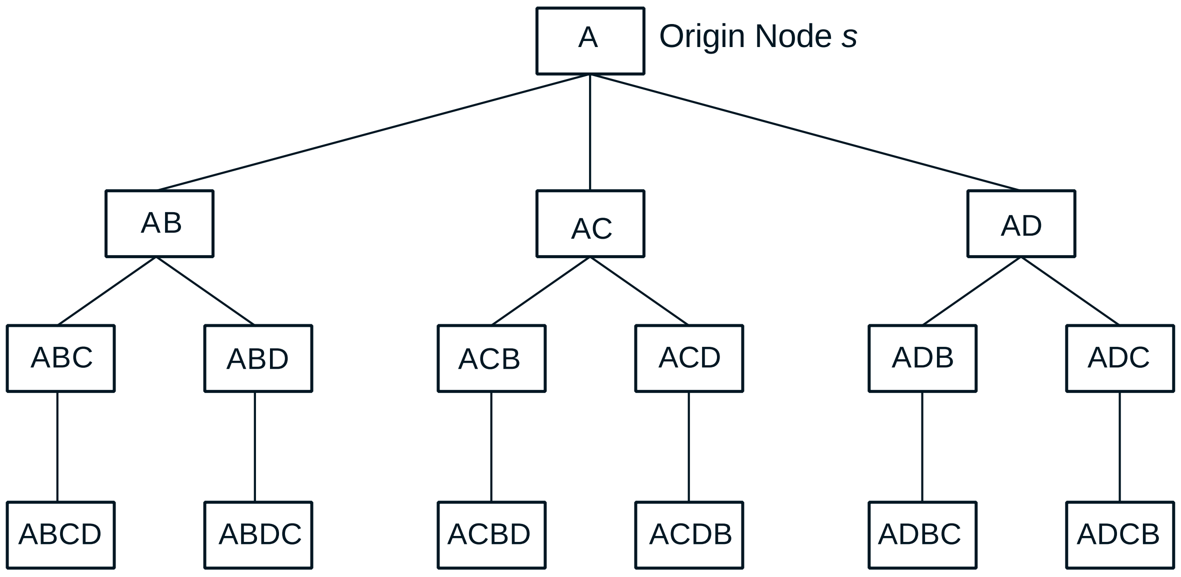
DISCRETE DETERMINISTIC PROBLEMS

- 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.

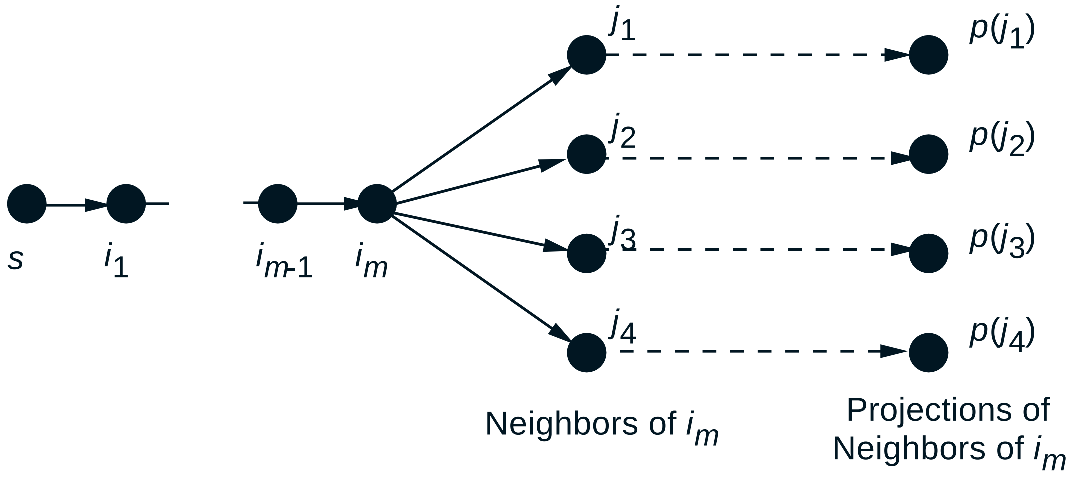
- **Example:** Traveling salesman problem. Find a minimum cost tour that goes exactly once through each of N cities.



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

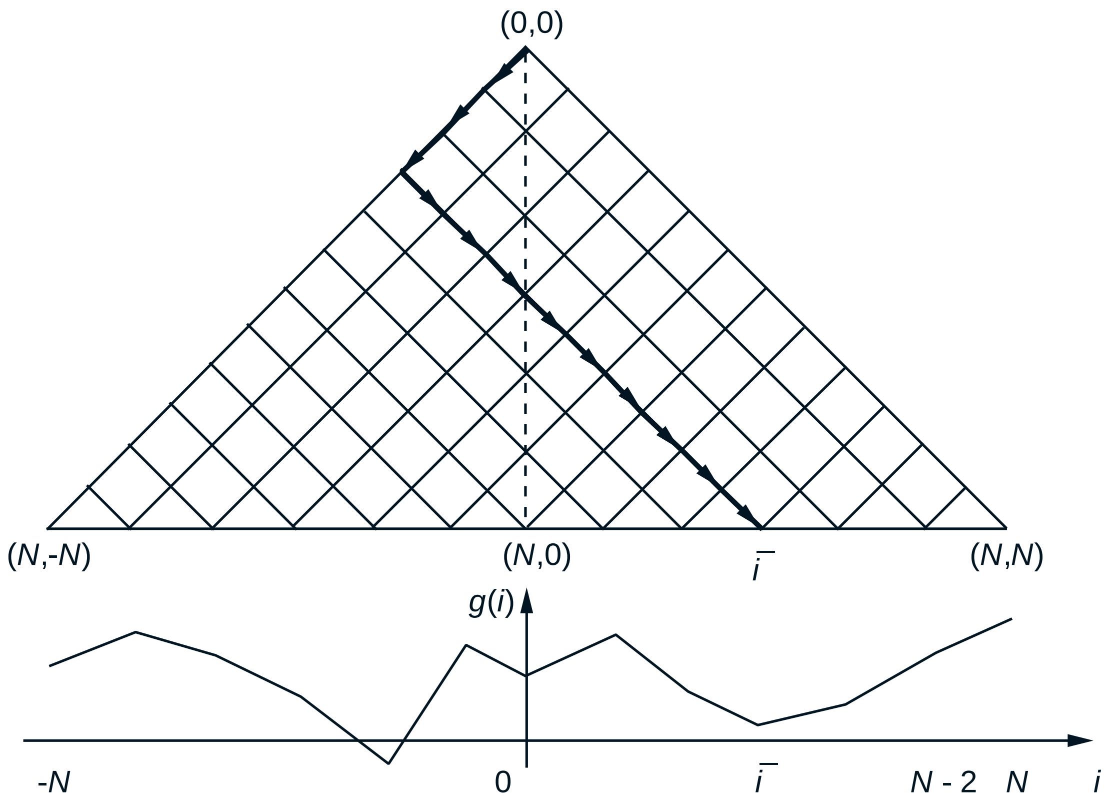
A CLASS OF GENERAL DISCRETE PROBLEMS

- 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 all nodes of its path have the same projection, i.e., 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 i_1.

- 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

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

- **Prop.:** 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 at i.

- **Proof:** Rollout deviates from the greedy path only when it discovers an improved path.

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 11

LECTURE OUTLINE

- More on rollout algorithms - Stochastic problems
- Simulation-based methods for rollout
- Approximations of rollout algorithms
- Rolling horizon approximations
- Discretization of continuous time
- Discretization of continuous space
- Other suboptimal approaches

ROLLOUT ALGORITHMS-STOCHASTIC CASE

- **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 the *Q-factor differences* $Q_k(x_k, u) - Q_k(x_k, \overline{u})$ directly. This may effect variance reduction of the simulation-induced error.

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

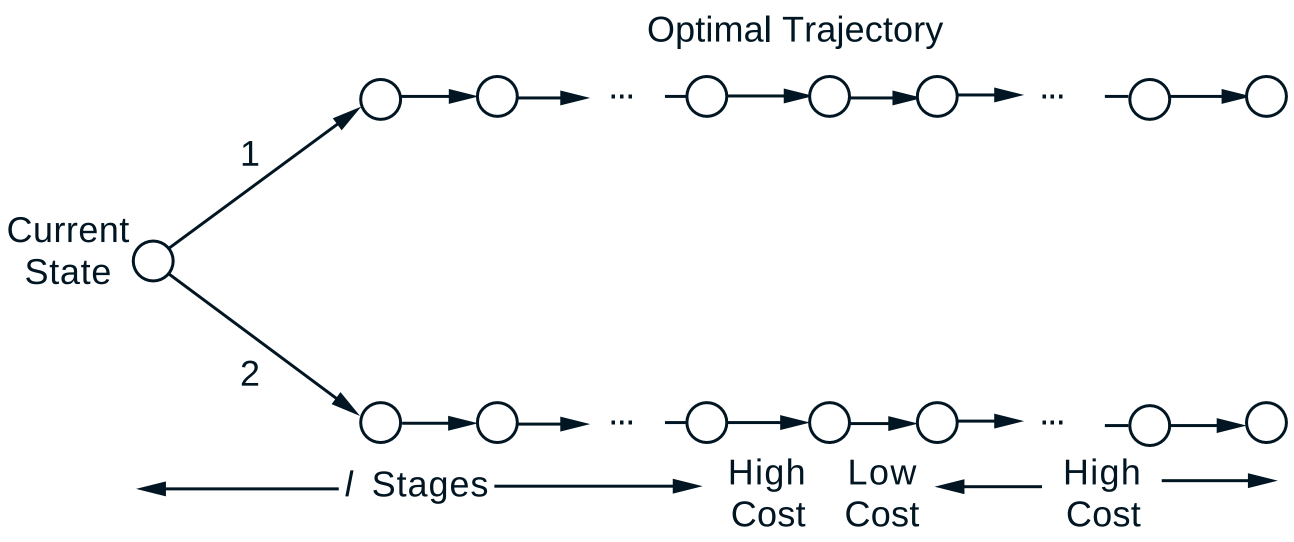
$$\tilde{Q}_k(x_k, u_k) = E\Big\{g_k(x_k, u_k, w_k) + \tilde{H}_{k+1}\big(f_k(x_k, u_k, w_k)\big)\Big\}$$

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 "training algorithm" or "least-squares fit."

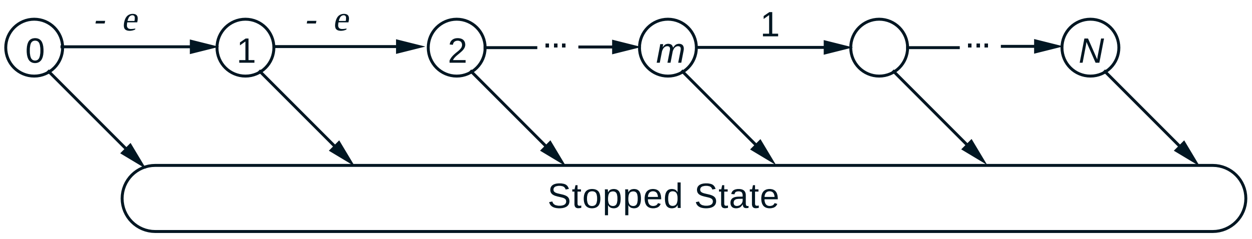
ROLLING HORIZON APPROACH

- 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 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 $0 < \epsilon << 1$, and the first state with continuation cost equal to 1 is state m. Then the optimal policy is to stop at state m, and the optimal cost is $-m\epsilon$.



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

MODEL PREDICTIVE CONTROL (MPC)

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

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

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

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

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.
- **Pitfall with discretizing continuous time:** The control constraint set may change a lot as we pass to the discrete-time approximation.
- **Example:** Let

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

with control constraint $u(t) \in \{-1, 1\}$. The reachable states after time δ are $x(t + \delta) = x(t) + u$, with $u \in [-\delta, \delta]$.

- Compare it with the reachable states after we discretize the system naively: $x(t + \delta) = x(t) + \delta u(t)$, with $u(t) \in \{-1, 1\}$.
- "Convexification effect" of continuous time: a discrete control constraint set in continuous-time differential systems, is equivalent to a continuous control constraint set when the system is looked at discrete times.

SPACE DISCRETIZATION I

- Given a discrete-time system with state space S, consider a finite subset $\overline{S}$; for example $\overline{S}$ could be a finite grid within a continuous state space S.
- Difficulty: $f(x,u,w) \notin \overline{S}$ for $x \in \overline{S}$.
- 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.
- Note application to finite-state POMDP (Partially Observed Markov Decision Problems)

SPACE 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 S) as long as the coefficients $\gamma_i(x)$ admit a meaningful interpretation that quantifies the degree of association of x with x_i.

OTHER SUBOPTIMAL APPROACHES

- **Minimize the DP equation error:** Approximate the optimal cost-to-go functions $J_k(x_k)$ with $\tilde{J}_k(x_k, r_k)$, where r_k is a parameter vector, chosen to minimize some form of error in the DP equations
  - Can be done sequentially going backwards in time (approximate J_k using an approximation of J_{k+1}).
- **Direct approximation of control policies:** For a subset of states x^i, $i = 1, \ldots, m$, 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 12

LECTURE OUTLINE

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

TYPES OF INFINITE HORIZON PROBLEMS

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

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

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

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

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

PREVIEW OF INFINITE HORIZON RESULTS

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

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

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

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

  - Bellman's Equation holds for all x:

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

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

STOCHASTIC SHORTEST PATH PROBLEMS

- Assume finite-state system: States $1, \ldots, n$ 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

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

  - 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 (Termination inevitable):** There exists integer m such that for every policy and initial state, there is positive probability that the termination state will be reached after no more 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 FUNCTIONS

- Let

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

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

- For any π and any initial state i

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

and similarly

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

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

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

and

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

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:

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

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

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

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

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

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

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

Take the minimum of both sides over π to obtain

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

Similarly, we have

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

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

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

$$J_N(x_0) \to J^*(x_0)$$

EXAMPLE

- 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

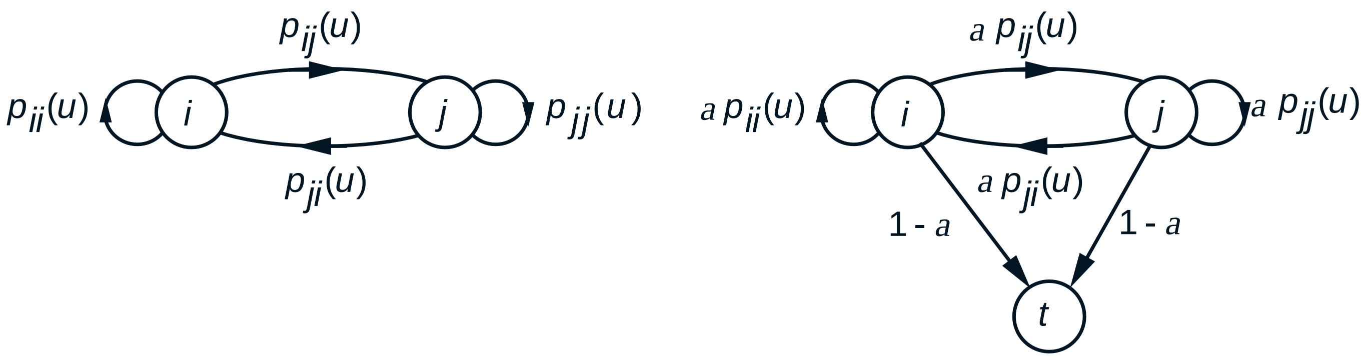
$$J^*(i) = \min_{u\in U(i)} \left[1 + \sum_{j=1}^n p_{ij}(u) J^*(j) \right], \quad i = 1,\ldots,n$$

- 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

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

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:

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 13

LECTURE OUTLINE

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

STOCHASTIC SHORTEST PATH PROBLEMS

- Assume finite-state system: States $1, \ldots, n$ 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 (Termination inevitable): There exists integer m such that for every policy and initial state, there is positive probability that the termination state will be reached after no more 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 value iteration

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

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 POLIC

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

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

- **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 $[O(n^3)]$
- For large n, value iteration may be preferable. (Typical case of a large linear system of equations, where an iterative method may be better than a direct solution method.)
- For VERY large n, exact methods cannot be applied, and approximations are needed. (We will discuss these later.)

POLICY ITERATION

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

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

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

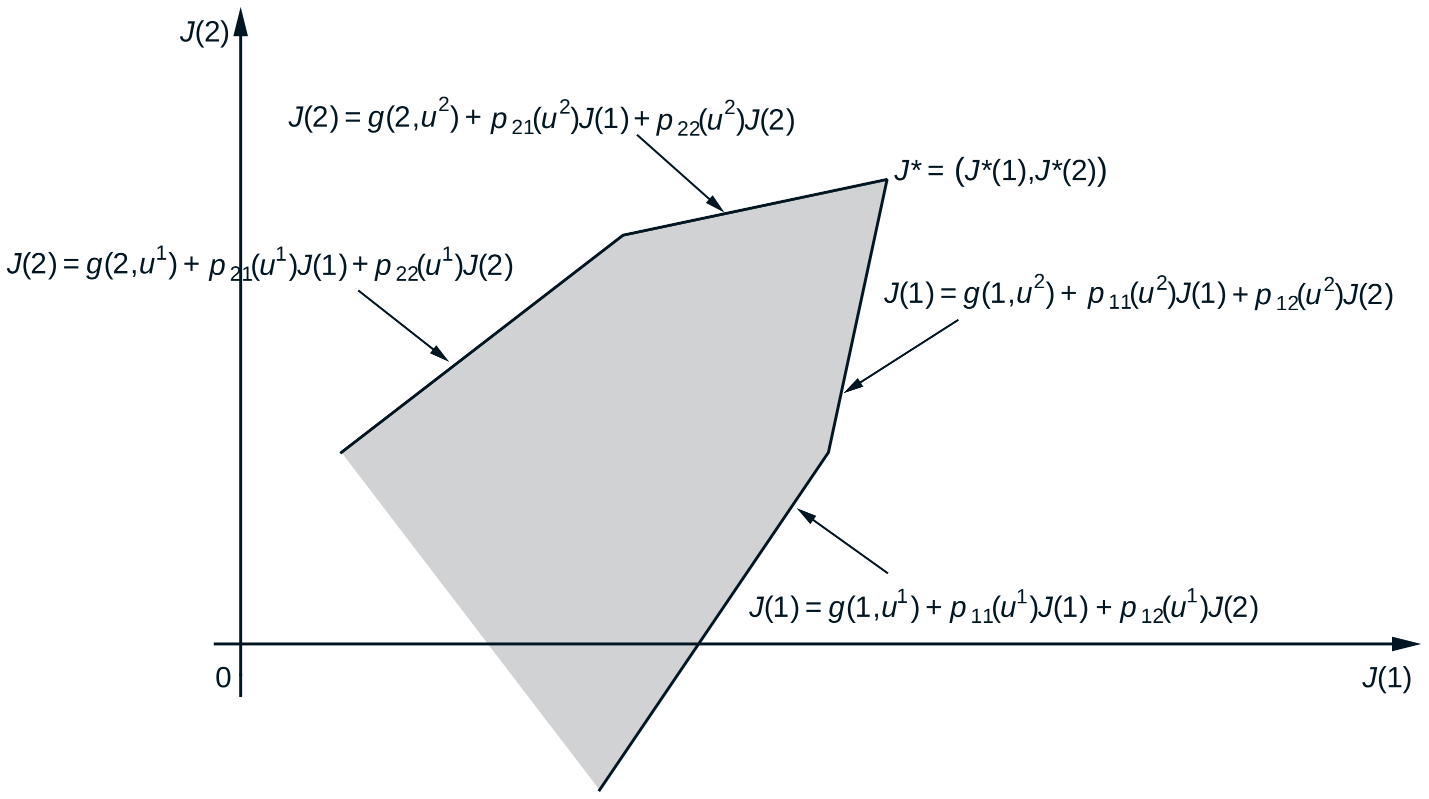
- **Proof:** If we use value iteration to generate a sequence of vectors $J_k = \big(J_k(1), \ldots, J_k(n)\big)$ starting with a J_0 such that

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

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

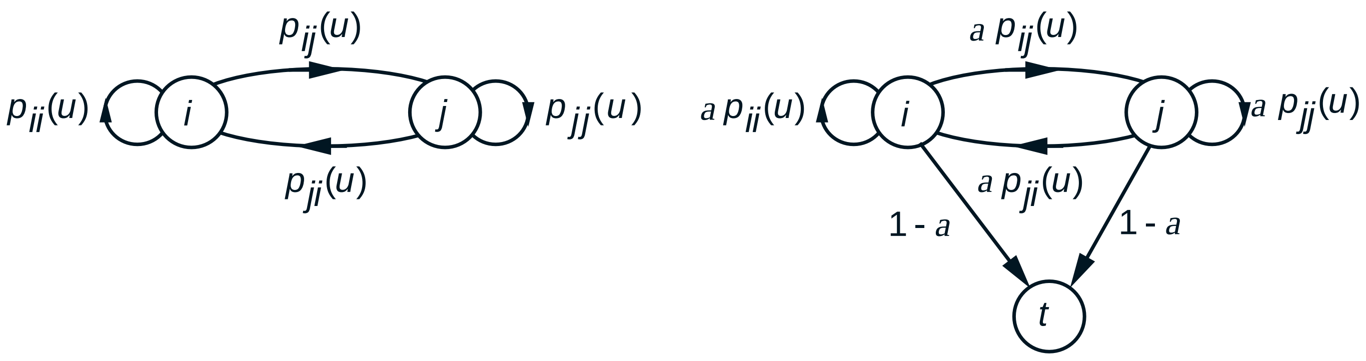


$J(2)$
$J(2) = g(2,u^2) + p_{21}(u^2)J(1) + p_{22}(u^2)J(2)$
$J^* = (J^*(1), J^*(2))$
$J(2) = g(2,u^1) + p_{21}(u^1)J(1) + p_{22}(u^1)J(2)$
$J(1) = g(1,u^2) + p_{11}(u^2)J(1) + p_{12}(u^2)J(2)$
$J(1) = g(1,u^1) + p_{11}(u^1)J(1) + p_{12}(u^1)J(2)$
0
$J(1)$

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

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

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

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

6.231 DYNAMIC PROGRAMMING

LECTURE 14

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

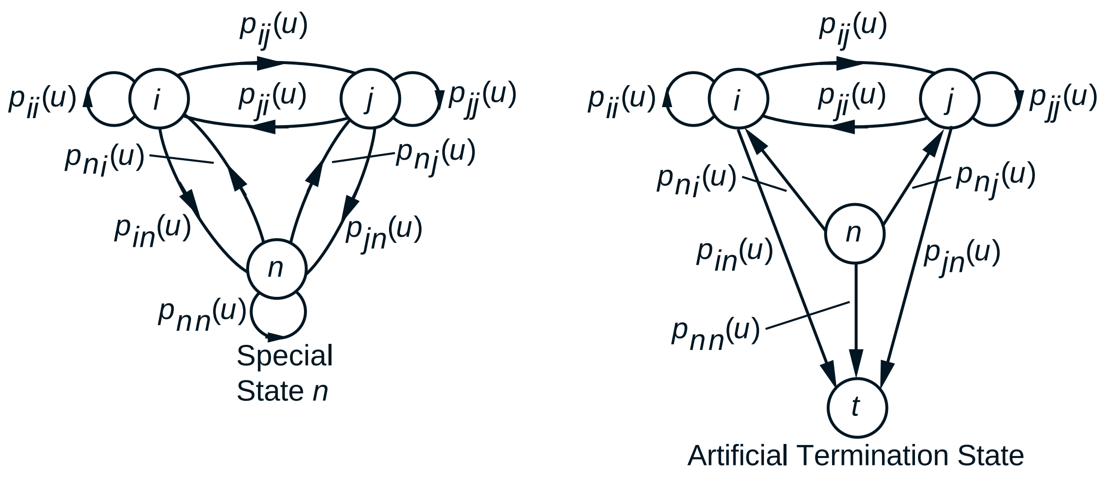
- Stationary system with finite number of states and controls
- Minimize over policies $\pi = \{\mu_0, \mu_1, ...\}$

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

- 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 $J^*(i) \le J^*(j)$]. 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 all initial states and all policies, n will be visited infinitely often (with probability 1).
- Divide the sequence of generated states into cycles marked by successive visits to n.
- Each of the cycles can be viewed as a state trajectory of a corresponding stochastic shortest path problem with n as the termination state.



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

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

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

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

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

with equality if μ is optimal.

- 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 $g(i,u) - \lambda^*$.
- Also: Optimal SSP Cost $= 0$.

BELLMAN'S EQUATION

- Let $h^*(i)$ the optimal cost of this SSP problem when starting at the nontermination states $i = 1, \ldots, n$. Then, $h^*(n) = 0$, and $h^*(1), \ldots, h^*(n)$ 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 (since Optimal SSP Cost $= 0$)

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

- 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

$$E\{\text{cost to reach } n \text{ from } i \text{ for the first time}\}$$
$$- E\{\text{cost if the stage cost were } \lambda^* \text{ and not } g(i,u)\}$$

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

EXAMPLE (CONTINUED)

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

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

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,

$$\big| J_k(i) - J_k^*(i) \big| \le \max_{j=1,\ldots,n} \big| J_0(j) - h^*(j) \big|, \qquad \forall \; i$$

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: (For the λ^k-SSP) Find for all i

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

- If $\lambda^{k+1} = \lambda^k$ and $h^{k+1}(i) = h^k(i)$ for all i, stop; otherwise, repeat with μ^{k+1} replacing μ^k.
- Result: For each k, we either have $\lambda^{k+1} < \lambda^k$ or we have policy improvement for the λ^k-SSP:

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

The algorithm terminates with an optimal policy.

6.231 DYNAMIC PROGRAMMING

LECTURE 15

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

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

- Two important formulas:

(1) Transition probabilities are specified by

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

(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 $\nu_i(u)$ (independently of j):

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

- The exponential distribution is memoryless. This implies that for a given policy, the system is a continuous-time Markov chain (the future depends on the past through the current state).

- Without the memoryless property, the Markov property holds only at the times of transition.

COST STRUCTURES

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

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

- There may be an extra "instantaneous" cost $\hat{g}(i,u)$ at the time of a transition (let's ignore this for the moment)

- Total discounted cost of $\pi = \{\mu_0, \mu_1, \ldots\}$ 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 CASE - COST CALCULATION

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

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

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 $E\{\text{1st transition cost}\}$, if u is applied at state i, is

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

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

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

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

COST CALCULATION (CONTINUED)

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

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

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

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

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

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

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

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

EQUIVALENCE TO AN SSP

- Similar to the discrete-time case, introduce a stochastic shortest path problem with an artificial termination state t

- Under control u, from state i the system moves to state j with probability $m_{ij}(u)$ and to the termination state t with probability $1 - \sum_{j=1}^{n} m_{ij}(u)$

- Bellman's equation: For $i = 1, \ldots, n$,

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

- 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

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

- 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:

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

MANUFACTURER EXAMPLE/AVERAGE COST

- 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 16

LECTURE OUTLINE

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

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

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

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

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

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

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

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

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

- For any stationary policy μ

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

“SHORTHAND” THEORY – A SUMMARY

- 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: $J^* = TJ^*$, $J_\mu = T_\mu J_\mu$
- 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

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

TWO KEY PROPERTIES

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

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

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

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

CONVERGENCE OF VALUE ITERATION

- If $J_0 \equiv 0$,

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

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

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

The tail portion satisfies

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

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

BELLMAN'S EQUATION

- The optimal cost function J^* satisfies Bellman's Eq., i.e. $J^* = T(J^*)$.

Proof: For all x and N,

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

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

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

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

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

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

THE CONTRACTION PROPERTY

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

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

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

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

$$J(x) - c \le J'(x) \le J(x) + c, \qquad \forall\ x$$

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

$$(TJ)(x) - \alpha c \le (TJ')(x) \le (TJ)(x) + \alpha c, \qquad \forall\ x$$

Hence

$$\big|(TJ)(x) - (TJ')(x)\big| \le \alpha c, \qquad \forall\ x.$$

Q.E.D.

IMPLICATIONS OF CONTRACTION PROPERTY

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

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

Proof: Use

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

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

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

for any bounded J.

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

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

NEC. AND SUFFICIENT OPT. CONDITION

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

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

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

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

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

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

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

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

COMPUTATIONAL METHODS

- Value iteration and variants
  - Gauss-Seidel version
  - Approximate value iteration
- Policy iteration and variants
  - Combination with value iteration
  - Modified policy iteration
  - Asynchronous policy iteration
- Linear programming

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

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

- Approximate linear programming: use in place of $J(i)$ a low-dim. basis function representation

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

and low-dim. LP (with many constraints)

6.231 DYNAMIC PROGRAMMING

LECTURE 17

LECTURE OUTLINE

- One-step lookahead and rollout for discounted problems
- Approximate policy iteration: Infinite state space
- Contraction mappings in DP
- Discounted problems: Countable state space with unbounded costs

ONE-STEP LOOKAHEAD POLICIES

- At state i use the control $\overline{\mu}(i)$ that attains the minimum in

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

where $\tilde{J}$ is some approximation to J^*.

- Assume that

$$T\tilde{J} \le \tilde{J} + \delta e,$$

for some scalar δ, where e is the unit vector. Then

$$J_{\overline{\mu}} \le T\tilde{J} + \frac{\alpha\delta}{1-\alpha}e \le \tilde{J} + \frac{\delta}{1-\alpha}e.$$

- Assume that

$$J^* - \epsilon e \le \tilde{J} \le J^* + \epsilon e,$$

for some scalar ϵ. Then

$$J_{\overline{\mu}} \le J^* + \frac{2\alpha\epsilon}{1-\alpha}e.$$

APPLICATION TO ROLLOUT POLICIES

- Let $\mu_1, \ldots, \mu_M$ be stationary policies, and let

$$\tilde{J}(i) = \min\bigl\{J_{\mu_1(i)}, \ldots, J_{\mu_M(i)}\bigr\}, \qquad \forall\ i.$$

- Then, for all i, and $m = 1, \ldots, M$, we have

$$\begin{aligned}(T\tilde{J})(i) &= \min_{u \in U(i)} \left[g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) \tilde{J}(j) \right] \\ &\leq \min_{u \in U(i)} \left[g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J_{\mu_m}(j) \right] \\ &\leq J_{\mu_m}(i)\end{aligned}$$

- Taking minimum over m,

$$(T\tilde{J})(i) \leq \tilde{J}(i), \qquad \forall\ i.$$

- Using the preceding slide result with $\delta = 0$,

$$J_{\overline{\mu}}(i) \leq \tilde{J}(i) = \min\bigl\{J_{\mu_1(i)}, \ldots, J_{\mu_M(i)}\bigr\}, \qquad \forall\ i,$$

i.e., the rollout policy $\overline{\mu}$ improves over each μ_m.

APPROXIMATE POLICY ITERATION

- 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 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 approximate policy iteration 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^*.

CONTRACTION MAPPINGS

- Given a real vector space Y with a norm $\|\cdot\|$ (i.e., $\|y\| \geq 0$ for all $y \in Y$, $\|y\| = 0$ if and only if $y = 0$, and $\|y + z\| \leq \|y\| + \|z\|$ for all $y, z \in Y$)
- A function $F : Y \mapsto Y$ is said to be a *contraction mapping* if for some $\rho \in (0, 1)$, we have

$$\|F(y) - F(z)\| \leq \rho\|y - z\|, \qquad \text{for all } y, z \in Y.$$

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

- For $m > 1$, we say that F is an *m-stage contraction* if F^m is a contraction.
- Important example: Let S be a set (e.g., state space in DP), $v : S \mapsto \Re$ be a positive-valued function. Let $B(S)$ be the set of all functions $J : S \mapsto \Re$ such that $J(s)/v(s)$ is bounded over s.
- We define a norm on $B(S)$, called the *weighted sup-norm*, by

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

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

CONTRACTION MAPPING FIXED-POINT TH.

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

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

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

$$\|F^k J - J^*\| \leq \rho^k \|J - J^*\|, \qquad k = 1, 2, \ldots.$$

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

A DP-LIKE CONTRACTION MAPPING I

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

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

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

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

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

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

where M is parameter set, and for each $\mu \in M$, F_μ is a contraction mapping from $B(S)$ to $B(S)$ with modulus ρ. Then F is a contraction mapping with modulus ρ.

A DP-LIKE CONTRACTION MAPPING II

- Let $S = \{1, 2, \ldots\}$, let M be a parameter set, and for each $\mu \in M$, let

$$(F_\mu J)(i) = b(i,\mu) + \sum_{j\in S} a(i,j,\mu)\, J(j), \qquad \forall\ i$$

- We have $F_\mu J \in B(S)$ for all $J \in B(S)$ provided $b_\mu \in B(S)$ and $V_\mu \in B(S)$, where

$$b_\mu = \big\{b(1,\mu), b(2,\mu), \ldots\big\},\ \ V_\mu = \big\{V(1,\mu), V(2,\mu), \ldots\big\}$$

$$V(i,\mu) = \sum_{j\in S} \big|a(i,j,\mu)\big|\, v(j), \qquad \forall\ i$$

- Consider the mapping F

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

We have $FJ \in B(S)$ for all $J \in B(S)$, provided $b \in B(S)$ and $V \in B(S)$, where

$$b = \big\{b(1), b(2), \ldots\big\}, \qquad V = \big\{V(1), V(2), \ldots\big\},$$

with $b(i) = \max_{\mu\in M} b(i,\mu)$ and $V(i) = \max_{\mu\in M} V(i,\mu)$.

DISCOUNTED DP - UNBOUNDED COST I

- State space $S = \{1, 2, \ldots\}$, transition probabilities $p_{ij}(u)$, cost $g(i, u)$.
- Weighted sup-norm

$$\|J\| = \max_{i \in S} \frac{|J(i)|}{v_i}$$

on $B(S)$: sequences $\{J(i)\}$ such that $\|J\| < \infty$.

- Assumptions:

(a) $G = \{G(1), G(2), \ldots\} \in B(S)$, where

$$G(i) = \max_{u \in U(i)} \big|g(i, u)\big|, \qquad \forall\ i$$

(b) $V = \{V(1), V(2), \ldots\} \in B(S)$, where

$$V(i) = \max_{u \in U(i)} \sum_{j \in S} p_{ij}(u)\, v_j, \qquad \forall\ i$$

(c) There exists an integer $m \geq 1$ and a scalar $\rho \in (0, 1)$ such that for every policy π,

$$\alpha^m \frac{\sum_{j \in S} P(x_m = j \mid x_0 = i, \pi)\, v_j}{v_i} \leq \rho, \qquad \forall\ i$$

DISCOUNTED DP - UNBOUNDED COST II

- Example: Let $v_i = i$ for all $i = 1, 2, \ldots$
- Assumption (a) is satisfied if the maximum expected absolute cost per stage at state i grows no faster than linearly with i.
- Assumption (b) states that the maximum expected next state following state i,

$$\max_{u \in U(i)} E\{j \mid i, u\},$$

also grows no faster than linearly with i.

- Assumption (c) is satisfied if

$$\alpha^m \sum_{j \in S} P(x_m = j \mid x_0 = i, \pi)\, j \leq \rho\, i, \qquad \forall\, i$$

It requires that for all π, the expected value of the state obtained m stages after reaching state i is no more than $\alpha^{-m} \rho\, i$.

- If there is bounded upward expected change of the state starting at i, there exists m sufficiently large so that Assumption (c) is satisfied.

DISCOUNTED DP - UNBOUNDED COST III

- Consider the DP mappings T_μ and T,

$$(T_\mu J)(i) = g\big(i,\mu(i)\big)+\alpha\sum_{j\in S} p_{ij}\big(\mu(i)\big)J(j), \qquad \forall\, i,$$

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

- **Proposition:** Under the earlier assumptions, T and T_μ map $B(S)$ into $B(S)$, and are m-stage contraction mappings with modulus ρ.

- The m-stage contraction properties can be used to essentially replicate the analysis for the case of bounded cost, and to show the standard results:
  - The value iteration method $J_{k+1} = TJ_k$ converges to the unique solution J^* of Bellman's equation $J = TJ$.
  - The unique solution J^* of Bellman's equation is the optimal cost function.
  - A stationary policy μ is optimal if and only if $T_\mu J^* = TJ^*$.

6.231 DYNAMIC PROGRAMMING

LECTURE 18

LECTURE OUTLINE

- Undiscounted problems
- Stochastic shortest path problems (SSP)
- Proper and improper policies
- Analysis and computational methods for SSP
- Pathologies of SSP

UNDISCOUNTED PROBLEMS

- System: $x_{k+1} = f(x_k, u_k, w_k)$
- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

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

- Shorthand notation for DP mappings

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

- For any stationary policy μ

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

- Neither T nor T_μ are contractions in general, but their monotonicity is helpful.
- SSP problems provide a “soft boundary” between the easy finite-state discounted problems and the hard undiscounted problems.
  - They share features of both.
  - Some of the nice theory is recovered because of the termination state.

SSP THEORY SUMMARY I

- As earlier, we have a cost-free term. state t, a finite number of states $1, \ldots, n$, and finite number of controls, but we will make weaker assumptions.
- Mappings T and T_μ (modified to account for termination state t):

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

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

- **Definition:** A stationary policy μ is called **proper**, if under μ, from every state i, there is a positive probability path that leads to t.
- **Important fact:** If μ is proper, T_μ is contraction with respect to some weighted max norm

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

- T is similarly a contraction if all μ are proper (the case discussed in the text, Ch. 7, Vol. I).

SSP THEORY SUMMARY II

- The theory can be pushed one step further. Assume that:

  (a) There exists at least one proper policy

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

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

SSP ANALYSIS I

- For a proper policy μ, J_μ is the unique fixed point of T_μ, and $T_\mu^k J \to J_\mu$ for all J (holds by the theory of Vol. I, Section 7.2)

- A stationary μ satisfying $J \geq T_\mu J$ for some J must be proper - true because

$$J \geq T_\mu^k J = P_\mu^k J + \sum_{m=0}^{k-1} P_\mu^m g_\mu$$

and some component of the term on the right blows up if μ is improper (by our assumptions).

- Consequence: T can have at most one fixed point.

Proof: If J and J' are two solutions, select μ and μ' such that $J = TJ = T_\mu J$ and $J' = TJ' = T_{\mu'} J'$. By preceding assertion, μ and μ' must be proper, and $J = J_\mu$ and $J' = J_{\mu'}$. Also

$$J = T^k J \leq T_{\mu'}^k J \to J_{\mu'} = J'$$

Similarly, $J' \leq J$, so $J = J'$.

SSP ANALYSIS II

- We now show that T has a fixed point, and also that policy iteration converges.
- Generate a sequence $\{\mu_k\}$ by policy iteration starting from a proper policy μ_0.
- μ_1 is proper and $J_{\mu_0} \geq J_{\mu_1}$ since

$$J_{\mu_0} = T_{\mu_0} J_{\mu_0} \geq T J_{\mu_0} = T_{\mu_1} J_{\mu_0} \geq T_{\mu_1}^k J_{\mu_0} \geq J_{\mu_1}$$

- Thus $\{J_{\mu_k}\}$ is nonincreasing, some policy μ will be repeated, with $J_\mu = T J_\mu$. So J_μ is a fixed point of T.
- Next show $T^k J \to J_\mu$ for all J, i.e., value iteration converges to the same limit as policy iteration. (Sketch: True if $J = J_\mu$, argue using the properness of μ to show that the terminal cost difference $J - J_\mu$ does not matter.)
- To show $J_\mu = J^*$, for any $\pi = \{\mu_0, \mu_1, \ldots\}$

$$T_{\mu_0} \cdots T_{\mu_{k-1}} J_0 \geq T^k J_0,$$

where $J_0 \equiv 0$. Take $\limsup$ as $k \to \infty$, to obtain $J_\pi \geq J_\mu$, so μ is optimal and $J_\mu = J^*$.

SSP ANALYSIS III

- If all policies are proper (the assumption of Section 7.1, Vol. I), T_μ and T are contractions with respect to a weighted sup norm.

Proof: Consider a new SSP problem where the transition probabilities are the same as in the original, but the transition costs are all equal to -1. Let $\hat{J}$ be the corresponding optimal cost vector. For all μ,

$$\hat{J}(i) = -1 + \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \hat{J}(j) \leq -1 + \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big) \hat{J}(j)$$

For $v_i = -\hat{J}(i)$, we have $v_i \geq 1$, and for all μ,

$$\sum_{j=1}^{n} p_{ij}\big(\mu(i)\big) v_j \leq v_i - 1 \leq \rho\, v_i, \qquad i = 1, \ldots, n,$$

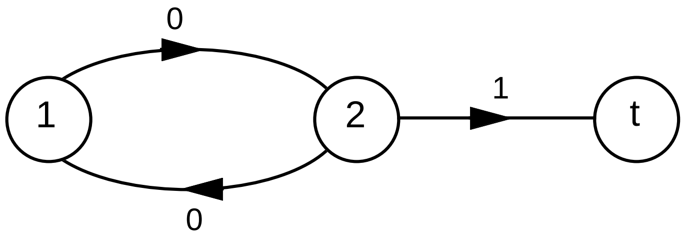
where

$$\rho = \max_{i=1,\ldots,n} \frac{v_i - 1}{v_i} < 1.$$

This implies contraction of T_μ and T by the results of the preceding lecture.

PATHOLOGIES I: DETERM. SHORTEST PATHS

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



- We have $J^*(1) = J^*(2) = 0$.
- Bellman's equation is

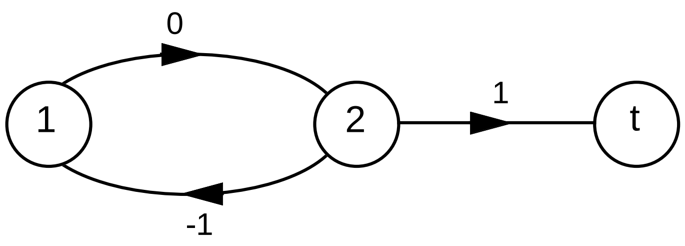
$$J(1) = J(2), \qquad J(2) = \min\big[J(1), 1\big].$$

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

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

PATHOLOGIES II: DETERM. SHORTEST PATHS

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



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

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

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

PATHOLOGIES III: THE BLACKMAILER

- Two states, state 1 and the termination state t.
- At state 1, choose a control $u \in (0, 1]$ (the blackmail amount demanded) at a cost $-u$, and move to t with probability u^2, or stay in 1 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

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

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

- Thus $J^*(1) = -\infty$, and there is no optimal stationary policy.
- It turns out that a nonstationary policy is optimal: demand $\mu_k(1) = \gamma/(k + 1)$ at time k, with $\gamma \in (0, 1/2)$.
  - Blackmailer requests diminishing amounts over time, which add to ∞.
  - The probability of the victim's refusal diminishes at a much faster rate, so the probability that the victim stays forever compliant is strictly positive.

6.231 DYNAMIC PROGRAMMING

LECTURE 19

LECTURE OUTLINE

- We begin a 6-lecture series on approximate DP for large/intractable problems.
- We will follow an updated/expanded version of Chapter 6, Vol. 2, which is posted at

http://web.mit.edu/dimitrib/www/dpchapter.html

- In this lecture we classify/overview the main approaches:
  - Rollout/Simulation-based single policy iteration (we will not discuss this further)
  - Approximation in value space (approximate policy iteration, Q-Learning, Bellman error approach, approximate LP)
  - Approximation in policy space (policy parametrization, gradient methods)
  - Problem approximation (simplification - aggregation - limited lookahead) - we will not discuss this much (see Vol. I, Ch. 6)

GENERAL ORIENTATION

- We will mainly adopt an n-state discounted model (the easiest case - but think of HUGE n).
- Extensions to SSP and average cost are possible (but more quirky). We will discuss them later.
- Other than manual/trial-and-error approaches (e.g., as in computer chess), the only other approaches are simulation-based. They are collectively known as "neuro-dynamic programming" or "reinforcement learning".
- Simulation is essential for large state spaces because of its (potential) computational complexity advantage in computing sums/expectations involving a very large number of terms.
- Simulation also comes in handy when an analytical model of the system is unavailable, but a simulation/computer model is possible.
- Simulation-based methods are of three types:
  - Rollout (we will not discuss further)
  - Approximation in value space
  - Approximation in policy space

APPROXIMATION IN POLICY SPACE

- A brief discussion; we will return to it at the end.
- We parameterize the set of policies by a vector $r = (r_1, \ldots, r_s)$ and we optimize the cost over r.
- Discounted problem example:
  - Each value of r defines a stationary policy, with cost starting at state i denoted by $J_i(r)$.
  - Use a gradient (or other) method to minimize over r

    $$\bar{J}(r) = \sum_{i=1}^{n} q(i) J_i(r),$$

    where $\big(q(1), \ldots, q(n)\big)$ is some probability distribution over the states.
- In a special case of this approach, the parameterization of the policies is indirect, through an approximate cost function.
  - A cost approximation architecture parameterized by r, defines a policy dependent on r via the minimization in Bellman's equation.

APPROX. IN VALUE SPACE - APPROACHES

- Policy evaluation/Policy improvement
  - Uses simulation algorithms to approximate the cost J_μ of the current policy μ
- Approximation of the optimal cost function J^*
  - Q-Learning: Use a simulation algorithm to approximate the optimal costs $J^*(i)$ or the Q-factors

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

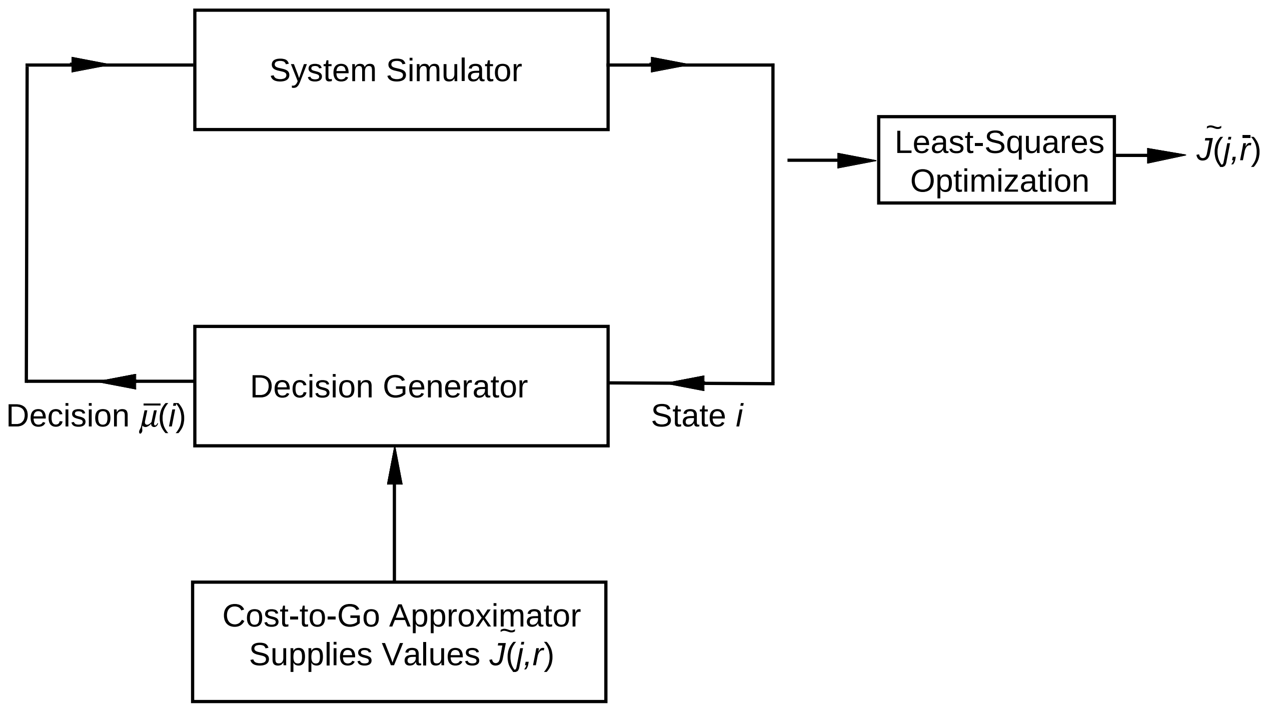
  - Bellman error approach: Find r to

$$\min_r E_i\Big\{ \big(\tilde{J}(i,r) - (T\tilde{J})(i,r)\big)^2 \Big\}$$

    where $E_i\{\cdot\}$ is taken with respect to some distribution
  - Approximate LP (discussed earlier - supplemented with clever schemes to overcome the large number of constraints issue)

POLICY EVALUATE/POLICY IMPROVE

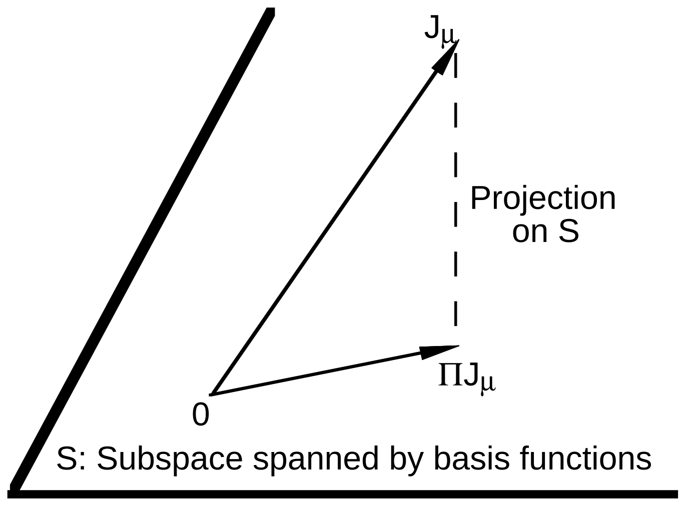
- An example



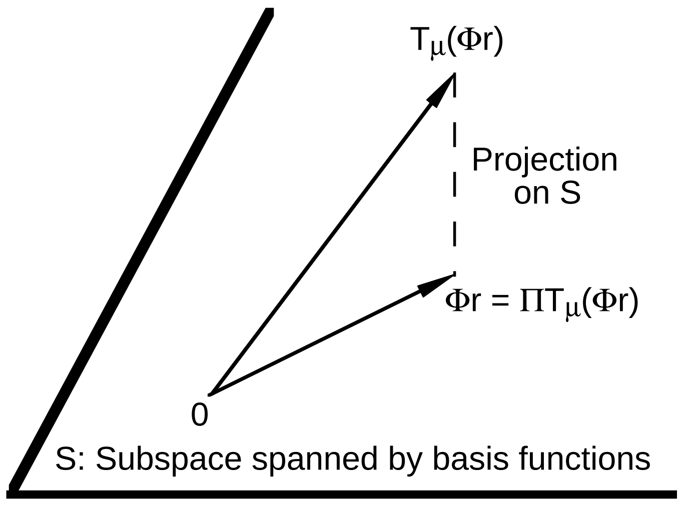
- The “least squares optimization” may be replaced by a different algorithm

POLICY EVALUATE/POLICY IMPROVE I

- Approximate the cost of the current policy by using a simulation method.
  - Direct policy evaluation - Cost samples generated by simulation, and optimization by least squares
  - Indirect policy evaluation - solving the projected equation $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm



Direct Mehod: Projection of cost vector J_μ



$T_\mu(\Phi r)$
Projection on S
$\Phi r = \Pi T_\mu(\Phi r)$
0
S: Subspace spanned by basis functions

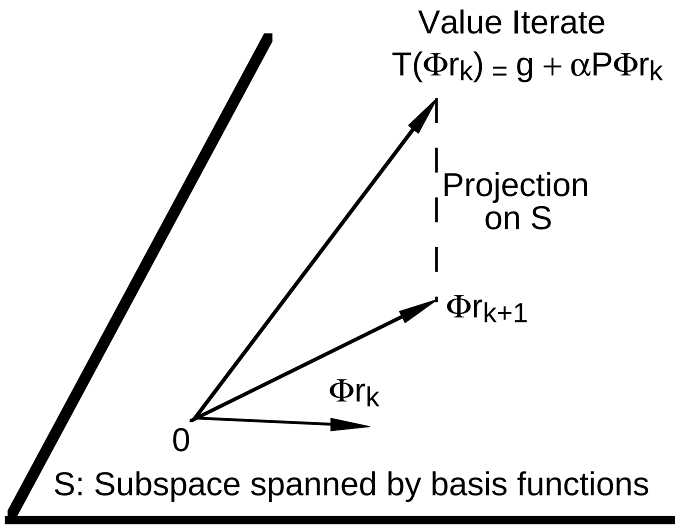
Indirect method: Solving a projected form of Bellman's equation

- Batch and incremental methods
- Regular and optimistic policy iteration

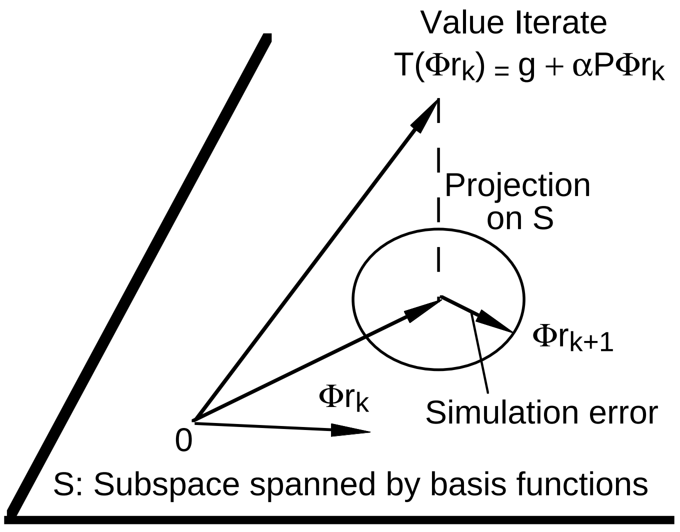
POLICY EVALUATE/POLICY IMPROVE II

- Projected equation methods are preferred and have rich theory
- TD(λ): Stochastic iterative algorithm for solving $\Phi r = \Pi T_\mu(\Phi r)$
- LSPE(λ): A simulation-based form of *projected value iteration*

$$\Phi r_{k+1} = \Pi T_\mu(\Phi r_k) + \text{ simulation noise}$$



Projected Value Iteration (PVI)



Least Squares Policy Evaluation (LSPE)

- LSTD(λ): Solves a simulation-based approximation $\Phi r = \hat{\Pi}\hat{T}_\mu(\Phi r)$ of the projected equation, using a linear system solver (e.g., Gaussian elimination/Matlab)

THEORETICAL BASIS

- If policies are approximately evaluated using an approximation architecture:

$$\max_i |\tilde{J}(i, r_k) - J_{\mu^k}(i)| \leq \delta, \qquad k = 0, 1, \ldots$$

- If policy improvement is also approximate,

$$\max_i |(T_{\mu^{k+1}}\tilde{J})(i, r_k) - (T\tilde{J})(i, r_k)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

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

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

- 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^*.

SIMULATION-BASED POLICY EVALUATION

- Suppose we can implement in a simulator the improved policy $\overline{\mu}$, and want to calculate $J_{\overline{\mu}}$ by simulation.

- Generate by simulation sample costs. Then:

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

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

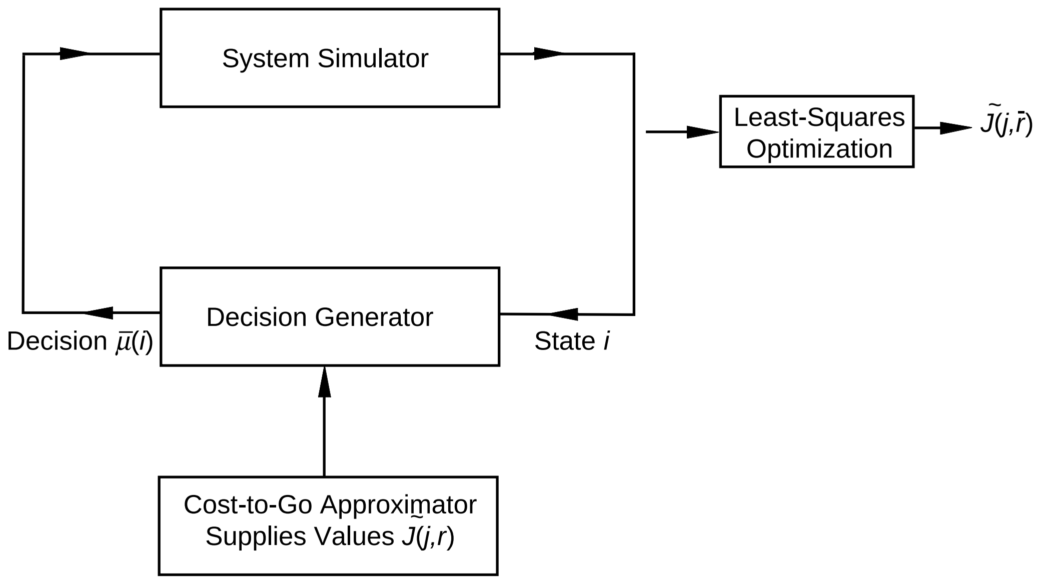
- Approximating well each $J_{\overline{\mu}}(i)$ is impractical for a large state space. Instead, a "compact representation" $\tilde{J}_{\overline{\mu}}(i,r)$ is used, where r is a tunable parameter vector.

- Direct approach: Calculate an optimal value r^* of r by a least squares fit

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

- Note that this is much easier when the architecture is linear - but this is not a requirement.

SIMULATION-BASED DIRECT APPROACH


System Simulator
Least-Squares Optimization
$\tilde{J}(j,\bar{r})$
Decision Generator
Decision $\overline{\mu}(i)$
State i
Cost-to-Go Approximator Supplies Values $\tilde{J}(j,r)$

- Simulator: Given a state-control pair (i, u), generates the next state j using system's transition probabilities under policy $\overline{\mu}$ currently evaluated
- Decision generator: Generates the control $\overline{\mu}(i)$ of the evaluated policy at the current state i
- Cost-to-go approximator: $\tilde{J}(j, r)$ used by the decision generator and corresponding to preceding policy (already evaluated in preceding iteration)
- Least squares optimizer: Uses cost samples $c(i, m)$ produced by the simulator and solves a least squares problem to approximate $\tilde{J}_{\overline{\mu}}(\cdot, \overline{r})$
- There are several important issues relating to the design of each block (to be discussed in the next lecture).

6.231 DYNAMIC PROGRAMMING

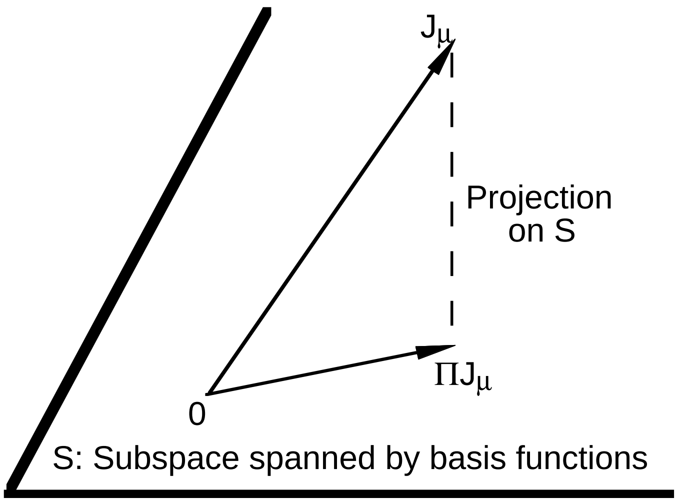
LECTURE 20

LECTURE OUTLINE

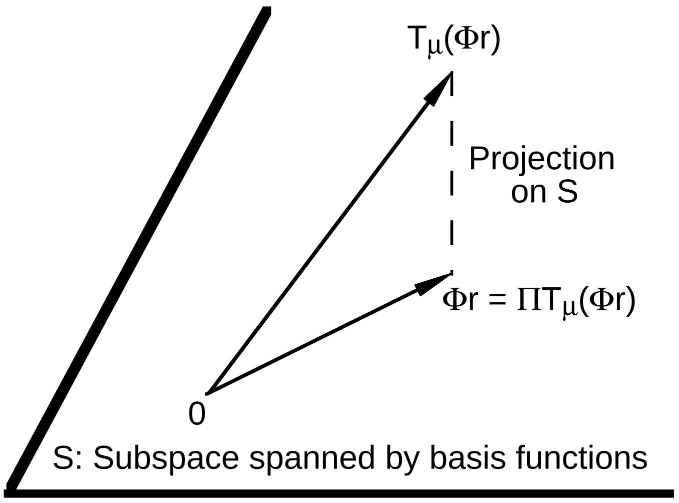
- Discounted problems - Approximate policy evaluation/policy improvement
- Exploration issues
- Direct approach - Least squares
- Batch and incremental gradient methods
- Implementation using TD
- Optimistic policy iteration

POLICY EVALUATE/POLICY IMPROVE

- Focus on policy evaluation: approximate the cost of the current policy by using a simulation method.
  - Direct policy evaluation - Cost samples generated by simulation, and optimization by least squares
  - Indirect policy evaluation - solving the projected equation $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm



Direct Mehod: Projection of cost vector J_μ


$T_\mu(\Phi r)$
Projection on S
$\Phi r = \Pi T_\mu(\Phi r)$
0
S: Subspace spanned by basis functions

Indirect method: Solving a projected form of Bellman's equation

THE ISSUE OF EXPLORATION

- To evaluate a policy μ, we need to generate cost samples using that policy - this biases the simulation by underrepresenting states that are unlikely to occur under μ.
- As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate.
- This seriously impacts the improved policy $\overline{\mu}$.
- This is known as *inadequate exploration* - a particularly acute difficulty when the randomness embodied in the transition probabilities is “relatively small” (e.g., a deterministic system).
- One possibility to guarantee adequate exploration: Frequently restart the simulation and ensure that the initial states employed form a rich and representative subset.
- Another possibility: Occasionally generating transitions that use a randomly selected control rather than the one dictated by the policy μ.
- Other methods, to be discussed later, use two Markov chains (one is the chain of the policy and is used to generate the transition sequence, the other is used to generate the state sequence).

APPROXIMATING Q-FACTORS

- The approach described so far for policy evaluation requires calculating expected values for all controls $u \in U(i)$ (and knowledge of $p_{ij}(u)$).
- Model-free alternative: Approximate Q-factors

$$\tilde{Q}(i,u,r) \approx \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J_\mu(j)\big)$$

and use for policy improvement the minimization

$$\overline{\mu}(i) = \arg\min_{u \in U(i)} \tilde{Q}(i,u,r)$$

- r is an adjustable parameter vector and $\tilde{Q}(i,u,r)$ is a parametric architecture, such as

$$\tilde{Q}(i,u,r) = \sum_{k=1}^{m} r_k \phi_k(i,u)$$

- Can use any method for constructing cost approximations, e.g., TD(λ).
- Use the Markov chain with states (i,u) - $p_{ij}(\mu(i))$ is the transition prob. to $(j,\mu(i))$, 0 to other (j,u').
- Major concern: Acutely diminished exploration.

DIRECT APPROACH

- Focus on a batch: an N-transition portion $(i_0, \dots, i_N)$ of a simulated trajectory
- We view the numbers

$$\sum_{t=k}^{N-1} \alpha^{t-k} g\big(i_t, \overline{\mu}(i_t), i_{t+1}\big), \qquad k = 0, \dots, N-1,$$

as cost samples, one per initial state $i_0, \dots, i_{N-1}$

- Least squares problem

$$\min_{\overline{r}} \frac{1}{2} \sum_{k=0}^{N-1} \left(\tilde{J}(i_k, \overline{r}) - \sum_{t=k}^{N-1} \alpha^{t-k} g\big(i_t, \overline{\mu}(i_t), i_{t+1}\big) \right)^2$$

- If $\tilde{J}(i_k, \overline{r})$ is linear in $\overline{r}$, this problem can be solved by matrix inversion.
- If $\tilde{J}(i_k, \overline{r})$ is linear in $\overline{r}$, or if the number of samples is very large, a gradient-type method may be preferable.

BATCH GRADIENT METHOD

- Gradient iteration

$$\overline{r} := \overline{r} - \gamma \sum_{k=0}^{N-1} \nabla \tilde{J}(i_k, \overline{r}) \left(\tilde{J}(i_k, \overline{r}) - \sum_{t=k}^{N-1} \alpha^{t-k} g\big(i_t, \overline{\mu}(i_t), i_{t+1}\big) \right)$$

- Important tradeoff:
  - In order to reduce simulation error and obtain cost samples for a representatively large subset of states, we must use a large N
  - To keep the work per gradient iteration small, we must use a small N

- To address the issue of size of N, small batches may be used and changed after one or more iterations.

- Then the method requires a diminishing stepsize for convergence.

- This slows down the convergence (which can generally be very slow for a gradient method).

- Theoretical convergence is guaranteed (with a diminishing stepsize) under reasonable conditions, but in practice this is not much of a guarantee.

INCREMENTAL GRADIENT METHOD

- Again focus on an N-transition portion $(i_0, \ldots, i_N)$ of a simulated trajectory.
- The batch gradient method processes the N transitions all at once, and updates $\overline{r}$ using the gradient iteration.
- The incremental method updates $\overline{r}$ a total of N times, once after each transition.
- After each transition (i_k, i_{k+1}) it uses only the portion of the gradient affected by that transition:
  - Evaluate the (single-term) gradient $\nabla\tilde{J}(i_k, \overline{r})$ at the current value of $\overline{r}$ (call it r_k).
  - Sum all the terms that involve the transition (i_k, i_{k+1}), and update r_k by making a correction along their sum:

$$\begin{aligned} r_{k+1} = & r_k - \gamma\Bigg(\nabla\tilde{J}(i_k, r_k)\tilde{J}(i_k, r_k) \\ & - \left(\sum_{t=0}^{k} \alpha^{k-t}\nabla\tilde{J}(i_t, r_t)\right) g\big(i_k, \overline{\mu}(i_k), i_{k+1}\big) \Bigg) \end{aligned}$$

INCREMENTAL GRADIENT - CONTINUED

- After N transitions, all the component gradient terms of the batch iteration are accumulated.
- BIG difference:
  - In the incremental method, $\overline{r}$ is changed while processing the batch – the (single-term) gradient $\nabla \tilde{J}(i_t, \overline{r})$ is evaluated at the most recent value of $\overline{r}$ [after the transition (i_t, i_{t+1})].
  - In the batch version these gradients are evaluated at the value of $\overline{r}$ prevailing at the beginning of the batch.
- Because $\overline{r}$ is updated at intermediate transitions within a batch (rather than at the end of the batch), the location of the end of the batch becomes less relevant.
- Can have very long batches - can have a single very long simulated trajectory and a single batch.
- The incremental version can be implemented more flexibly, converges much faster in practice.
- Interesting convergence analysis (beyond our scope - see Bertsekas and Tsitsiklis, NDP book, also paper in SIAM J. on Optimization, 2000)

TEMPORAL DIFFERENCES - TD(1)

- A mathematically equivalent implementation of the incremental method.
- It uses *temporal difference* (TD for short)

$$q_k = g\big(i_k, \overline{\mu}(i_k), i_{k+1}\big) + \alpha \tilde{J}(i_{k+1}, \overline{r}) - \tilde{J}(i_k, \overline{r}),\ \ k \le N-2$$

$$q_{N-1} = g\big(i_{N-1}, \overline{\mu}(i_{N-1}), i_N\big) - \tilde{J}(i_{N-1}, \overline{r})$$

- Following the transition (i_k, i_{k+1}), set

$$r_{k+1} = r_k + \gamma_k q_k \sum_{t=0}^{k} \alpha^{k-t} \nabla \tilde{J}(i_t, r_t)$$

- This algorithm is known as TD(1). In the important linear case $\tilde{J}(i, r) = \phi(i)'r$, it becomes

$$r_{k+1} = r_k + \gamma_k q_k \sum_{t=0}^{k} \alpha^{k-t} \phi(i_t)$$

- A variant of TD(1) is TD(λ), $\lambda \in [0, 1]$. It sets

$$r_{k+1} = r_k + \gamma_k q_k \sum_{t=0}^{k} (\alpha\lambda)^{k-t} \phi(i_t)$$

OPTIMISTIC POLICY ITERATION

- We have assumed so far is that the least squares optimization must be solved completely for $\overline{r}$.
- An alternative, known as *optimistic policy iteration*, is to solve this problem approximately and replace policy μ with policy $\overline{\mu}$ after only a few simulation samples.
- Extreme possibility is to replace μ with $\overline{\mu}$ at the end of each state transition: After state transition (i_k, i_{k+1}), set

$$r_{k+1} = r_k + \gamma_k q_k \sum_{t=0}^{k} (\alpha\lambda)^{k-t} \nabla \tilde{J}(i_t, r_t),$$

and simulate next transition (i_{k+1}, i_{k+2}) using $\overline{\mu}(i_{k+1})$, the control of the new policy.

- For $\lambda = 0$, we obtain (the popular) optimistic TD(0), which has the simple form

$$r_{k+1} = r_k + \gamma_k q_k \nabla \tilde{J}(i_k, r_k)$$

- Optimistic policy iteration can exhibit fascinating and counterintuitive behavior (see the NDP book by Bertsekas and Tsitsiklis, Section 6.4.2).

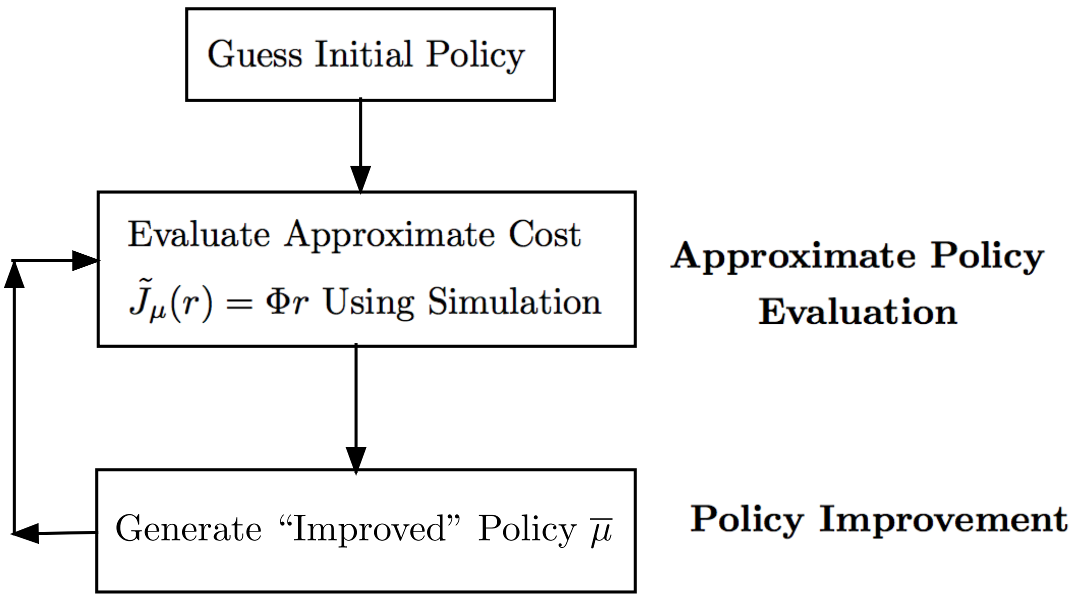
6.231 DYNAMIC PROGRAMMING

LECTURE 21

LECTURE OUTLINE

- Discounted problems - Approximate policy evaluation/policy improvement
- Indirect approach - The projected equation
- Contraction properties - Error bounds
- Matrix form of the projected equation
- Simulation-based implementation (LSTD)
- Iterative methods (LSPE)

POLICY EVALUATION/POLICY IMPROVEMENT


Guess Initial Policy
Evaluate Approximate Cost $\tilde{J}_\mu(r) = \Phi r$ Using Simulation
Approximate Policy Evaluation
Generate "Improved" Policy $\overline{\mu}$
Policy Improvement

- Linear cost function approximation

$$\tilde{J}(r) = \Phi r$$

where Φ is full rank $n \times s$ matrix with columns the basis functions, and ith row denoted $\phi(i)'$.

- Policy "improvement"

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \phi(j)' r\big)$$

- Indirect methods find Φr by solving a projected equation.

WEIGHTED EUCLIDEAN PROJECTIONS

- Consider a weighted Euclidean norm

$$\|J\|_v = \sqrt{\sum_{i=1}^{n} v_i \big(J(i)\big)^2},$$

where v is a vector of positive weights $v_1, \ldots, v_n$.

- Let Π denote the projection operation onto

$$S = \{\Phi r \mid r \in \Re^s\}$$

with respect to this norm, i.e., for any $J \in \Re^n$,

$$\Pi J = \Phi r^*$$

where

$$r^* = \arg\min_{r \in \Re^s} \|J - \Phi r\|_v^2$$

THE PROJECTED BELLMAN EQUATION

- For a fixed policy μ to be evaluated, consider the corresponding mapping T:

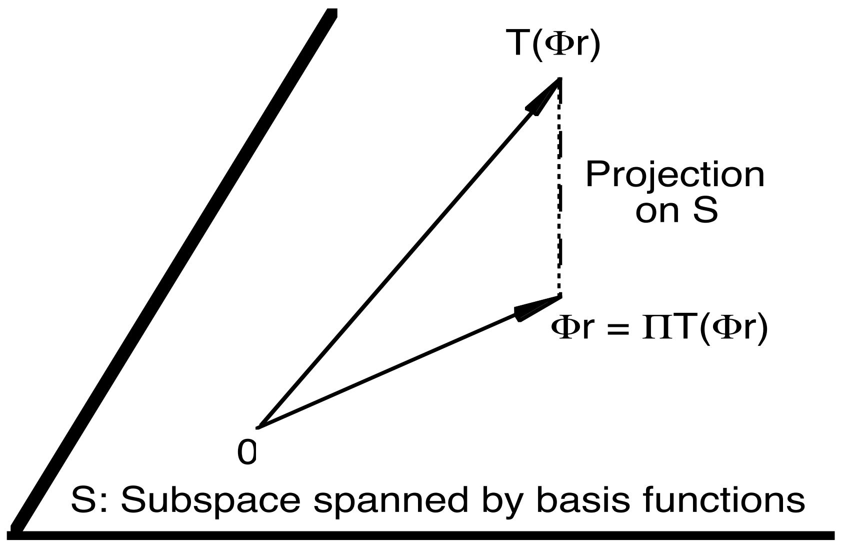
$$(TJ)(i) = \sum_{i=1}^{n} p_{ij}\big(g(i,j)+\alpha J(j)\big), \qquad i = 1,\ldots,n,$$

or more compactly,

$$TJ = g + \alpha PJ$$

- The solution J_μ of Bellman's equation $J = TJ$ is approximated by the solution of

$$\Phi r = \Pi T(\Phi r)$$



Indirect method: Solving a projected form of Bellman's equation

KEY QUESTIONS AND RESULTS

- Does the projected equation have a solution?
- Under what conditions is the mapping ΠT a contraction, so ΠT has unique fixed point?
- Assuming ΠT has unique fixed point Φr^*, how close is Φr^* to J_μ?
- **Assumption:** P has a single recurrent class and no transient states, i.e., it has steady-state probabilities that are positive

$$\xi_j = \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} P(i_k = j \mid i_0 = i) > 0$$

- **Proposition:** ΠT is contraction of modulus α with respect to the weighted Euclidean norm $\| \cdot \|_\xi$, where $\xi = (\xi_1, \ldots, \xi_n)$ is the steady-state probability vector. The unique fixed point Φr^* of ΠT satisfies

$$\|J_\mu - \Phi r^*\|_\xi \le \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi$$

ANALYSIS

- Important property of the projection Π on S with weighted Euclidean norm $\|\cdot\|_v$. For all $J \in \Re^n$, $\overline{J} \in S$, the *Pythagorean Theorem* holds:

$$\|J - \overline{J}\|_v^2 = \|J - \Pi J\|_v^2 + \|\Pi J - \overline{J}\|_v^2$$

- Proof: Geometrically, $(J - \Pi J)$ and $(\Pi J - \overline{J})$ are orthogonal in the scaled geometry of the norm $\|\cdot\|_v$, where two vectors $x, y \in \Re^n$ are orthogonal if $\sum_{i=1}^n v_i x_i y_i = 0$. Expand the quadratic in the RHS below:

$$\|J - \overline{J}\|_v^2 = \|(J - \Pi J) + (\Pi J - \overline{J})\|_v^2$$

- The Pythagorean Theorem implies that the projection is *nonexpansive*, i.e.,

$$\|\Pi J - \Pi \bar{J}\|_v \leq \|J - \bar{J}\|_v, \qquad \text{for all } J, \bar{J} \in \Re^n.$$

To see this, note that

$$\begin{aligned}\big\|\Pi(J - \overline{J})\big\|_v^2 &\leq \big\|\Pi(J - \overline{J})\big\|_v^2 + \big\|(I - \Pi)(J - \overline{J})\big\|_v^2 \\ &= \|J - \overline{J}\|_v^2\end{aligned}$$

PROOF OF CONTRACTION PROPERTY

- Lemma: We have

$$\|Pz\|_\xi \le \|z\|_\xi, \qquad z \in \Re^n$$

- Proof of lemma: Let p_{ij} be the components of P. For all $z \in \Re^n$, we have

$$\begin{aligned}\|Pz\|_\xi^2 &= \sum_{i=1}^n \xi_i \left(\sum_{j=1}^n p_{ij} z_j\right)^2 \le \sum_{i=1}^n \xi_i \sum_{j=1}^n p_{ij} z_j^2 \\ &= \sum_{j=1}^n \sum_{i=1}^n \xi_i p_{ij} z_j^2 = \sum_{j=1}^n \xi_j z_j^2 = \|z\|_\xi^2,\end{aligned}$$

where the inequality follows from the convexity of the quadratic function, and the next to last equality follows from the defining property $\sum_{i=1}^n \xi_i p_{ij} = \xi_j$ of the steady-state probabilities.

- Using the lemma, the nonexpansiveness of Π, and the definition $TJ = g + \alpha PJ$, we have

$$\|\Pi TJ - \Pi T\bar{J}\|_\xi \le \|TJ - T\bar{J}\|_\xi = \alpha\|P(J-\bar{J})\|_\xi \le \alpha\|J-\bar{J}\|_\xi$$

for all $J, \bar{J} \in \Re^n$. Hence ΠT is a contraction of modulus α.

PROOF OF ERROR BOUND

- Let Φr^* be the fixed point of ΠT. We have

$$\|J_\mu - \Phi r^*\|_\xi \le \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi.$$

Proof: We have

$$\begin{aligned}
\|J_\mu - \Phi r^*\|_\xi^2 &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \big\|\Pi J_\mu - \Phi r^*\big\|_\xi^2 \\
&= \|J_\mu - \Pi J_\mu\|_\xi^2 + \big\|\Pi T J_\mu - \Pi T(\Phi r^*)\big\|_\xi^2 \\
&\le \|J_\mu - \Pi J_\mu\|_\xi^2 + \alpha^2 \|J_\mu - \Phi r^*\|_\xi^2,
\end{aligned}$$

where the first equality uses the Pythagorean Theorem, the second equality holds because J_μ is the fixed point of T and Φr^* is the fixed point of ΠT, and the inequality uses the contraction property of ΠT. From this relation, the result follows.

- Note: The factor $1/\sqrt{1-\alpha^2}$ in the RHS can be replaced by a factor that is smaller and computable. See
H. Yu and D. P. Bertsekas, "New Error Bounds for Approximations from Projected Linear Equations," Report LIDS-P-2797, MIT, July 2008.

MATRIX FORM OF PROJECTED EQUATION

- Its solution is the vector $J = \Phi r^*$, where r^* solves the problem

$$\min_{r \in \Re^s} \big\| \Phi r - (g + \alpha P \Phi r^*) \big\|_\xi^2 .$$

- Setting to 0 the gradient with respect to r of this quadratic, we obtain

$$\Phi' \Xi \big(\Phi r^* - (g + \alpha P \Phi r^*) \big) = 0,$$

where Ξ is the diagonal matrix with the steady-state probabilities $\xi_1, \ldots, \xi_n$ along the diagonal.

- Equivalently,

$$C r^* = d,$$

where

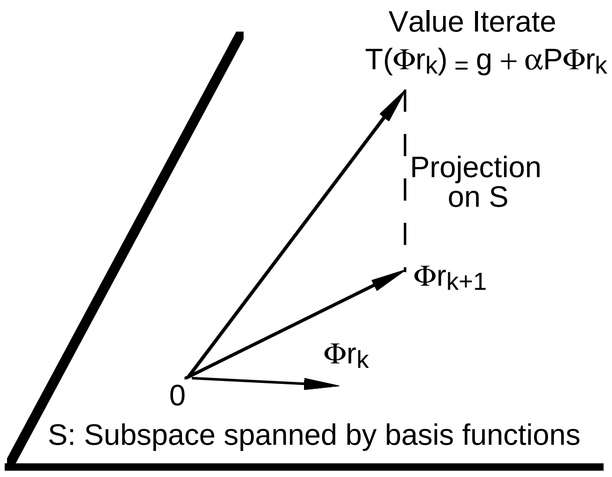
$$C = \Phi' \Xi (I - \alpha P) \Phi, \qquad d = \Phi' \Xi g.$$

PROJECTED EQUATION: SOLUTION METHODS

- Matrix inversion: $r^* = C^{-1}d$
- Projected Value Iteration (PVI) method:

$$\Phi r_{k+1} = \Pi T(\Phi r_k) = \Pi\big(g + \alpha P \Phi r_k\big)$$

Converges to r^* because ΠT is a contraction.



- PVI can be written as:

$$r_{k+1} = \arg\min_{r \in \Re^s} \big\|\Phi r - (g + \alpha P \Phi r_k)\big\|_\xi^2$$

By setting to 0 the gradient with respect to r,

$$\Phi' \Xi \big(\Phi r_{k+1} - (g + \alpha P \Phi r_k)\big) = 0,$$

which yields

$$r_{k+1} = r_k - (\Phi' \Xi \Phi)^{-1}(C r_k - d)$$

SIMULATION-BASED IMPLEMENTATIONS

- Key idea: Calculate simulation-based approximations based on k samples

$$C_k \approx C, \qquad d_k \approx d$$

- Matrix inversion $r^* = C^{-1}d$ is approximated by

$$\hat{r}_k = C_k^{-1} d_k$$

This is the LSTD (Least Squares Temporal Differences) Method.

- PVI method $r_{k+1} = r_k - (\Phi'\Xi\Phi)^{-1}(Cr_k - d)$ is approximated by

$$r_{k+1} = r_k - D_k^{-1}(C_k r_k - d_k)$$

where

$$D_k \approx \Phi'\Xi\Phi$$

This is the LSPE (Least Squares Policy Evaluation) Method.

- Key fact: C_k, d_k, and D_k can be computed with low-dimensional linear algebra (of order s; the number of basis functions).

SIMULATION MECHANICS

- We generate an infinitely long trajectory $(i_0, i_1, \ldots)$ of the Markov chain, so states i and transitions (i, j) appear with long-term frequencies ξ_i and p_{ij}.

- After generating the transition (i_t, i_{t+1}), we compute the row $\phi(i_t)'$ of Φ and the cost component $g(i_t, i_{t+1})$.

- We form

$$C_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\big(\phi(i_t) - \alpha\phi(i_{t+1})\big)' \approx \Phi'\Xi(I - \alpha P)\Phi$$

$$d_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) g(i_t, i_{t+1}) \approx \Phi'\Xi g$$

Also in the case of LSPE

$$D_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \approx \Phi'\Xi\Phi$$

- Convergence proof is simple: Use the law of large numbers.

6.231 DYNAMIC PROGRAMMING

LECTURE 22

LECTURE OUTLINE

- More on projected equation approach
- Q-Learning
- Q-Learning with Function Approximation
- Aggregation

REVIEW: PROJECTED BELLMAN EQUATION

- For a fixed policy μ to be evaluated, consider the corresponding mapping T:

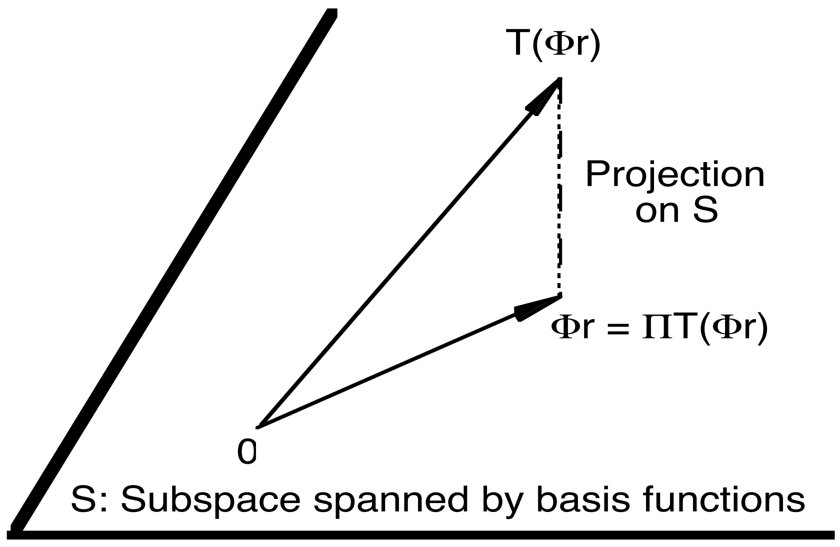
$$(TJ)(i) = \sum_{i=1}^{n} p_{ij}\big(g(i,j)+\alpha J(j)\big), \qquad i = 1,\ldots,n,$$

or more compactly,

$$TJ = g + \alpha PJ$$

- The solution J_μ of Bellman's equation $J = TJ$ is approximated by the solution of

$$\Phi r = \Pi T(\Phi r)$$



Indirect method: Solving a projected form of Bellman's equation

MULTISTEP METHODS

• Introduce a multistep version of Bellman's equation $J = T^{(\lambda)}J$, where for $\lambda \in [0,1)$,

$$T^{(\lambda)} = (1-\lambda)\sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

• Note that T^t is a contraction with modulus α^t, with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where ξ is the steady-state probability vector of the Markov chain.

• From this it follows that $T^{(\lambda)}$ is a contraction with modulus

$$\alpha_\lambda = (1-\lambda)\sum_{t=0}^{\infty} \alpha^{t+1}\lambda^t = \frac{\alpha(1-\lambda)}{1-\alpha\lambda}$$

• T^t and $T^{(\lambda)}$ have the same fixed point J_μ and

$$\|J_\mu - \Phi r_\lambda^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha_\lambda^2}}\,\|J_\mu - \Pi J_\mu\|_\xi$$

where Φr_λ^* is the fixed point of $\Pi T^{(\lambda)}$.

• The fixed point Φr_λ^* depends on λ.

• Note that $\alpha_\lambda \downarrow 0$ as $\lambda \uparrow 1$, so error bound (and the quality of approximation) improves as $\lambda \uparrow 1$.

MULTISTEP PROJECTED EQUATION

- The projected Bellman equation is

$$\Phi r = \Pi T^{(\lambda)}(\Phi r)$$

- In matrix form: $C^{(\lambda)} r = d^{(\lambda)}$, where

$$C^{(\lambda)} = \Phi' \Xi \big(I - \alpha P^{(\lambda)}\big)\Phi, \qquad d^{(\lambda)} = \Phi' \Xi g^{(\lambda)},$$

with

$$P^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P^{\ell+1}, \quad g^{(\lambda)} = \sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P^\ell g$$

- The LSTD(λ) method is

$$\big(C_k^{(\lambda)}\big)^{-1} d_k^{(\lambda)},$$

where $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ are simulation-based approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$.

- The LSPE(λ) method is

$$r_{k+1} = r_k - \gamma D_k^{-1}\big(C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big)$$

where D_k is a simulation-based approx. to $\Phi' \Xi \Phi$

- TD(λ) is a simplified version of LSPE(λ) where $D_k = I$, and simpler approximations to C and d are used. It is more properly viewed as a stochastic iteration for solving the projected equation.

MORE ON MULTISTEP METHODS

- The simulation process to obtain $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ is similar to the case $\lambda = 0$ (single simulation trajectory $i_0, i_1, \ldots$ but the formulas are more complicated)

$$C_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \sum_{m=t}^{k} \alpha^{m-t} \lambda^{m-t} \big(\phi(i_m) - \alpha \phi(i_{m+1})$$

$$d_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \sum_{m=t}^{k} \alpha^{m-t} \lambda^{m-t} g_{i_m}$$

- Note the λ-tradeoff:
  - As $\lambda \uparrow 1$ $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ contain more "simulation noise", so more samples are needed for a close approximation of r_λ, the solution of the projected equation
  - The error bound $\|J_\mu - \Phi r_\lambda\|_\xi$ is becomes smaller

- In the context of approximate policy iteration, we can use optimistic versions (few samples between policy updates). Also exploration is always an issue.

- Many different versions ... see on-line chapter.

Q-LEARNING I

- Q-learning has two motivations:
  - Dealing with multiple policies simultaneously
  - Using a model-free approach [no need to know $p_{ij}(u)$, only be able to simulate them]
- The Q-factors are defined by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ (i,u)$$

- Since $J^* = TJ^*$, we have $J^*(i) = \min_{u \in U(i)} Q^*(i,u)$ so the Q factors solve the equation

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

- $Q^*(i,u)$ can be shown to be the unique solution of this equation. **Reason:** This is Bellman's equation for a system whose states are the original states $1, \ldots, n$, together with all the pairs (i,u).
- Value iteration: For all (i,u)

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

Q-LEARNING II

- Use any probabilistic mechanism to select sequence of pairs (i_k, u_k) [all pairs (i, u) are chosen infinitely often], and for each k, select j_k according to $p_{i_k j}(u_k)$.

- At each k, Q-learning algorithm updates $Q(i_k, u_k)$ according to

$$
\begin{aligned}
Q(i_k, u_k) := & \big(1 - \gamma_k(i_k, u_k)\big) Q(i_k, u_k) \\
& + \gamma_k(i_k, u_k) \left(g(i_k, u_k, j_k) + \alpha \min_{u' \in U(j_k)} Q(j_k, u') \right)
\end{aligned}
$$

- Stepsize $\gamma_k(i_k, u_k)$ must converge to 0 at proper rate (e.g., like $1/k$).

- **Important mathematical point:** In the Q-factor version of Bellman's equation the order of expectation and minimization is reversed relative to the ordinary cost version of Bellman's equation:

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

- Q-learning can be shown to converge to true/exact Q-factors (a sophisticated proof).

- **Major drawback:** The large number of pairs (i, u) - no function approximation is used.

Q-FACTOR APROXIMATIONS

- Introduce basis function approximation for Q-factors:

$$\tilde{Q}(i,u,r) = \phi(i,u)'r$$

- We can use approximate policy iteration and LSPE/LSTD/TD for policy evaluation (exploration issue is acute)

- Optimistic policy iteration methods are frequently used on a heuristic basis.

- **Example:** Generate infinite trajectory $\{(i_k,u_k) \mid k=0,1,\ldots\}$

- At iteration k, given r_k and state/control (i_k,u_k):

  (1) Simulate next transition (i_k,i_{k+1}) using the transition probabilities $p_{i_kj}(u_k)$.

  (2) Generate control u_{k+1} from

$$u_{k+1} = \arg \min_{u \in U(i_{k+1})} \tilde{Q}(i_{k+1},u,r_k)$$

  (3) Update the parameter vector via

$$r_{k+1} = r_k - (\text{LSPE or TD-like correction})$$

- Unclear validity. There is a solid basis for an important but very special case: optimal stopping (see the on-line chapter)

PROBLEM APPROXIMATION - AGGREGATION

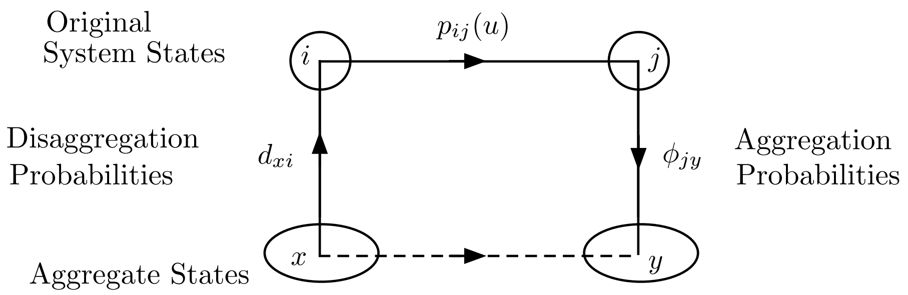
- Another major idea in ADP is to approximate the cost-to-go function of the problem with the cost-to-go function of a simpler problem. The simplification is often ad-hoc/problem dependent.

- Aggregation is a (semi-)systematic approach for problem approximation. Main elements:
  - Introduce a few "aggregate" states, viewed as the states of an "aggregate" system
  - Define transition probabilities and costs of the aggregate system, by 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 approximate 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).

AGGREGATION/DISAGGREGATION PROBS

- The aggregate system transition probabilities are defined via two (somewhat arbitrary) choices:
- For each aggregate state x and original system state i, the disaggregation probability d_{xi}
  - This may be roughly interpreted as the "degree to which i is representative of x."
  - In the hard aggregation example (assuming all states that belong to aggregate state/subset x are "equally representative") $d_{xi} = 1/|x|$ for each state i that belongs to aggregate state/subset x, where $|x|$ is the cardinality (number of states) of x.
- For each original system state j and aggregate state y, the aggregation probability ϕ_{jy}
  - This may be roughly interpreted as the "degree of membership of j in the aggregate state y."
  - In the hard aggregation example, $\phi_{jy} = 1$ if state j belongs to aggregate state/subset y.



$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}$$

AGGREGATE TRANSITION PROBABILITIES

- The transition probability from aggregate state x to aggregate state y under control u

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}$$

- The expected transition cost is

$$\hat{g}(x,u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)g(i,u,j)$$

- The optimal cost function of the aggregate problem, denoted $\hat{J}$, is obtained as the unique solution of Bellman's equation

$$\hat{J}(x) = \min_{u \in U} \left[\hat{g}(x,u) + \alpha \sum_{y} \hat{p}_{xy}(u)\hat{J}(y) \right], \qquad \forall\, x$$

- The optimal cost function J^* of the original problem is approximated by $\tilde{J}$ given by

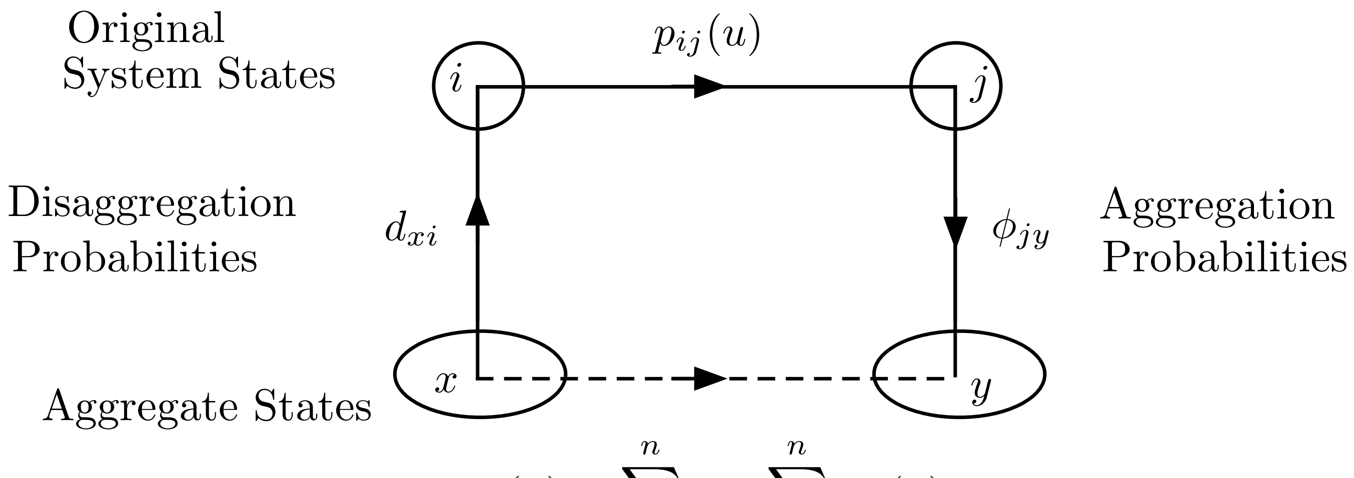
$$\tilde{J}(j) = \sum_{y} \phi_{jy}\hat{J}(y), \qquad \forall\, j$$

Equivalent Form (Similar to Projected Equation): $\tilde{J} = \Phi r$, where $r = DT(\Phi r)$ where the rows of D and Φ are the disaggr. and aggr. probs.

- May solve aggregate problem by Q-learning or approximate policy iteration with policy evaluation step done by aggregation

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)
  - A coarse grid is well-suited for continuous space problems
  - Important example is POMDP, where the state space of the sufficient statistic is the space of beliefs (a continuous space)



$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}$$

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

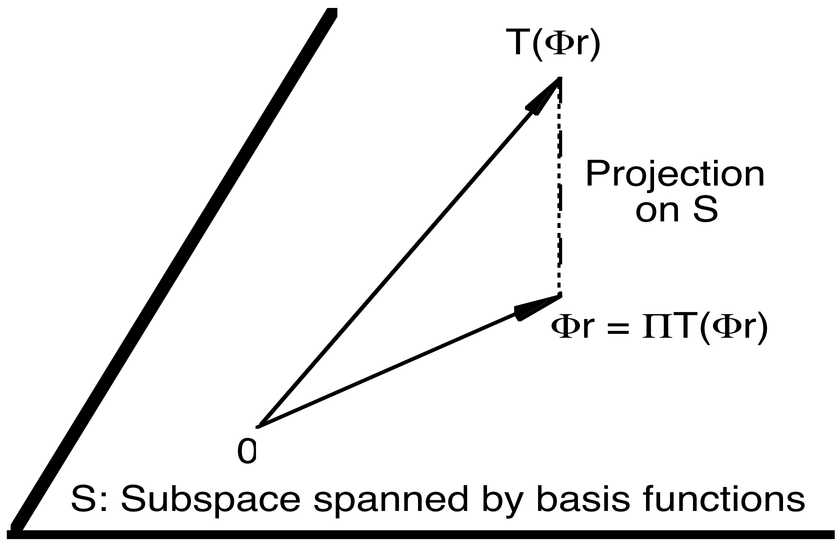
- More on projected equation methods/policy evaluation
- Stochastic shortest path problems
- Average cost problems
- Generalization - Two Markov Chain methods
- LSTD-like methods - Use to enhance exploration

REVIEW: PROJECTED BELLMAN EQUATION

- For fixed policy μ to be evaluated, the solution of Bellman's equation $J = TJ$ is approximated by the solution of

$$\Phi r = \Pi T(\Phi r)$$

whose solution is in turn obtained using a simulation-based method such as LSPE(λ), LSTD(λ), or TD(λ).



Indirect method: Solving a projected form of Bellman's equation

- These ideas apply to other (linear) Bellman equations, e.g., for SSP and average cost.
- Key Issue: Construct framework where ΠT [or at least $\Pi T^{(\lambda)}$] is a contraction.

STOCHASTIC SHORTEST PATHS

- Introduce approximation subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

and for a given proper policy, Bellman's equation and its projected version

$$J = TJ = g + PJ, \qquad \Phi r = \Pi T(\Phi r)$$

Also its λ-version

$$\Phi r = \Pi T^{(\lambda)}(\Phi r), \qquad T^{(\lambda)} = (1-\lambda)\sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

- **Question:** What should be the norm of projection?
- **Speculation based on discounted case:** It should be a weighted Euclidean norm with weight vector $\xi = (\xi_1, \ldots, \xi_n)$, where ξ_i should be some type of long-term occupancy probability of state i (which can be generated by simulation).
- But what does "long-term occupancy probability of a state" mean in the SSP context?
- How do we generate infinite length trajectories given that termination occurs with prob. 1?

SIMULATION TRAJECTORIES FOR SSP

- We envision simulation of trajectories up to termination, followed by restart at state i with some fixed probabilities $q_0(i) > 0$.
- Then the "long-term occupancy probability of a state" of i is proportional to

$$q(i) = \sum_{t=0}^{\infty} q_t(i), \qquad i = 1, \ldots, n,$$

where

$$q_t(i) = P(i_t = i), \qquad i = 1, \ldots, n,\ t = 0, 1, \ldots$$

- We use the projection norm

$$\|J\|_q = \sqrt{\sum_{i=1}^{n} q(i)\big(J(i)\big)^2}$$

[Note that $0 < q(i) < \infty$, but q is not a prob. distribution.]

- We can show that $\Pi T^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_\xi$ (see the next slide).
- LSTD(λ), LSPE(λ), and TD(λ) are possible.

CONTRACTION PROPERTY FOR SSP

- We have $q = \sum_{t=0}^{\infty} q_t$ so

$$q'P = \sum_{t=0}^{\infty} q_t' P = \sum_{t=1}^{\infty} q_t' = q' - q_0'$$

or

$$\sum_{i=1}^{n} q(i) p_{ij} = q(j) - q_0(j), \qquad \forall\ j$$

- To verify that ΠT is a contraction, we show that there exists $\beta < 1$ such that $\|Pz\|_q^2 \leq \beta \|z\|_q^2$ for all $z \in \Re^n$.

- For all $z \in \Re^n$, we have

$$\begin{aligned}\|Pz\|_q^2 &= \sum_{i=1}^{n} q(i) \left(\sum_{j=1}^{n} p_{ij} z_j \right)^2 \leq \sum_{i=1}^{n} q(i) \sum_{j=1}^{n} p_{ij} z_j^2 \\ &= \sum_{j=1}^{n} z_j^2 \sum_{i=1}^{n} q(i) p_{ij} = \sum_{j=1}^{n} \big(q(j) - q_0(j) \big) z_j^2 \\ &= \|z\|_q^2 - \|z\|_{q_0}^2 \leq \beta \|z\|_q^2\end{aligned}$$

where

$$\beta = 1 - \min_j \frac{q_0(j)}{q(j)}$$

AVERAGE COST PROBLEMS

- Consider a single policy to be evaluated, with single recurrent class, no transient states, and steady-state probability vector $\xi = (\xi_1, \ldots, \xi_n)$.
- The average cost, denoted by η, is independent of the initial state

$$\eta = \lim_{N \to \infty} \frac{1}{N} E\left\{ \sum_{k=0}^{N-1} g\big(x_k, x_{k+1}\big) \;\Big|\; x_0 = i \right\}, \quad \forall\, i$$

- Bellman's equation is $J = FJ$ with

$$FJ = g - \eta e + PJ$$

where e is the unit vector $e = (1, \ldots, 1)$.

- The projected equation and its λ-version are

$$\Phi r = \Pi F(\Phi r), \qquad \Phi r = \Pi F^{(\lambda)}(\Phi r)$$

- A problem here is that F is not a contraction with respect to any norm (since $e = Pe$).
- However, $\Pi F^{(\lambda)}$ turns out to be a contraction with respect to $\|\cdot\|_\xi$ assuming that e does not belong to S and $\lambda > 0$ [the case $\lambda = 0$ is exceptional, but can be handled - see the text].
- LSTD(λ), LSPE(λ), and TD(λ) are possible.

GENERALIZATION/UNIFICATION

- Consider approximate solution of $x = T(x)$, where

$$T(x) = Ax + b, \qquad A \text{ is } n \times n, \quad b \in \Re^n$$

by solving the projected equation $y = \Pi T(y)$, where Π is projection on a subspace of basis functions (with respect to some Euclidean norm).

- We will generalize from DP to the case where A is arbitrary, subject only to

$$I - \Pi A: \text{ invertible}$$

- Benefits of generalization:
  - Unification/higher perspective for TD methods in approximate DP
  - An extension to a broad new area of applications, where a DP perspective may be helpful
- Challenge: Dealing with less structure
  - Lack of contraction
  - Absence of a Markov chain

GENERALIZED PROJECTED EQUATION

- Let Π be projection with respect to

$$\|x\|_\xi = \sqrt{\sum_{i=1}^{n} \xi_i x_i^2}$$

where $\xi \in \Re^n$ is a probability distribution with positive components.

- If r^* is the solution of the projected equation, we have $\Phi r^* = \Pi(A\Phi r^* + b)$ or

$$r^* = \arg\min_{r\in\Re^s} \sum_{i=1}^{n} \xi_i \left(\phi(i)'r - \sum_{j=1}^{n} a_{ij}\phi(j)'r^* - b_i \right)^2$$

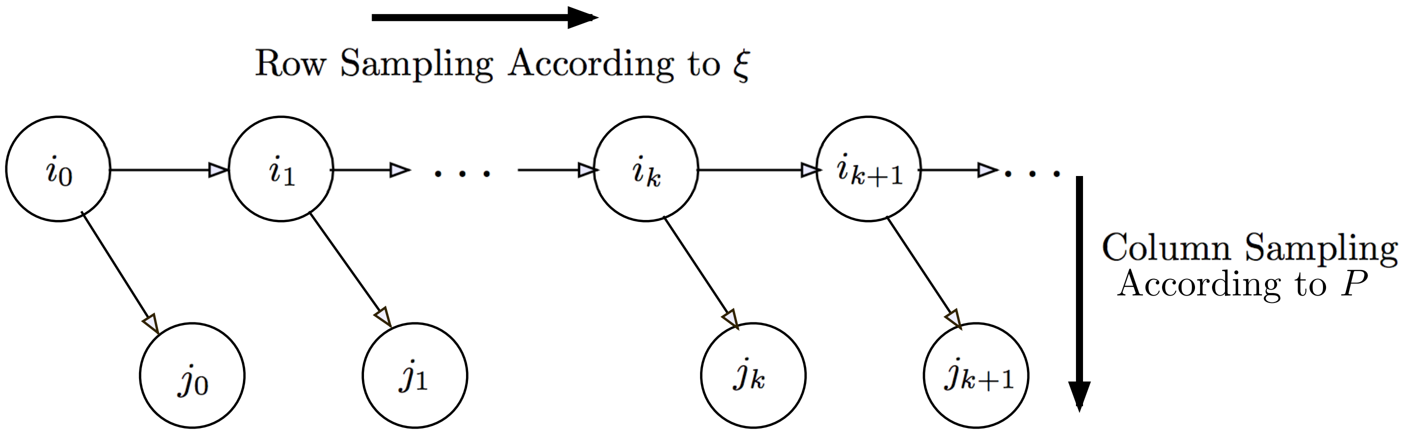
where $\phi(i)'$ denotes the ith row of the matrix Φ.

- Optimality condition/equivalent form:

$$\sum_{i=1}^{n} \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^{n} a_{ij}\phi(j) \right)' r^* = \sum_{i=1}^{n} \xi_i \phi(i) b_i$$

- The two expected values can be approximated by simulation.

SIMULATION MECHANISM


Row Sampling According to ξ
i_0
i_1
i_k
i_{k+1}
j_0
j_1
j_k
j_{k+1}
Column Sampling According to P

- Row sampling: Generate sequence $\{i_0, i_1, \ldots\}$ according to ξ, i.e., relative frequency of each row i is ξ_i

- Column sampling: Generate $\{(i_0, j_0), (i_1, j_1), \ldots\}$ according to some transition probability matrix P with

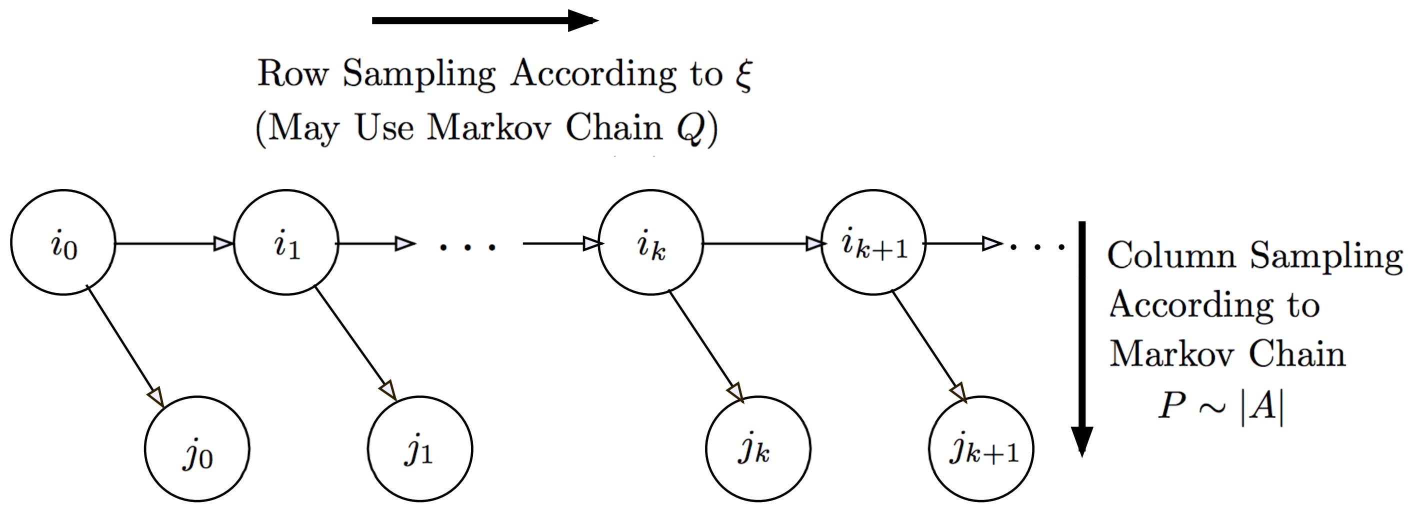
$$p_{ij} > 0 \qquad \text{if} \qquad a_{ij} \neq 0,$$

i.e., for each i, the relative frequency of (i, j) is p_{ij}

- Row sampling may be done using a Markov chain with transition matrix Q (unrelated to P)

- Row sampling may also be done without a Markov chain - just sample rows according to some known distribution ξ (e.g., a uniform)

ROW AND COLUMN SAMPLING


Row Sampling According to ξ
(May Use Markov Chain Q)
i_0
i_1
i_k
i_{k+1}
j_0
j_1
j_k
j_{k+1}
Column Sampling According to Markov Chain $P \sim |A|$

- Row sampling $\sim$ State Sequence Generation in DP. Affects:
  - The projection norm.
  - Whether ΠA is a contraction.
- Column sampling $\sim$ Transition Sequence Generation in DP.
  - Can be totally unrelated to row sampling. Affects the sampling/simulation error.
  - "Matching" P with $|A|$ is beneficial (has an effect like in importance sampling).
- Independent row and column sampling allows exploration at will! Resolves the exploration problem that is critical in approximate policy iteration.

LSTD-LIKE METHOD

- Optimality condition/equivalent form of projected equation

$$\sum_{i=1}^{n} \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^{n} a_{ij} \phi(j) \right)' r^* = \sum_{i=1}^{n} \xi_i \phi(i) b_i$$

- The two expected values are approximated by row and column sampling (batch $0 \to t$).
- We solve the linear equation

$$\sum_{k=0}^{t} \phi(i_k) \left(\phi(i_k) - \frac{a_{i_k j_k}}{p_{i_k j_k}} \phi(j_k) \right)' r_t = \sum_{k=0}^{t} \phi(i_k) b_{i_k}$$

- We have $r_t \to r^*$, regardless of ΠA being a contraction (by law of large numbers; see next slide).
- An LSPE-like method is also possible, but requires that ΠA is a contraction.
- Under the assumption $\sum_{j=1}^{n} |a_{ij}| \leq 1$ for all i, there are conditions that guarantee contraction of ΠA; see the paper by Bertsekas and Yu, "Projected Equation Methods for Approximate Solution of Large Linear Systems," 2009, or the expanded version of Chapter 6, Vol. 2.

JUSTIFICATION W/ LAW OF LARGE NUMBERS

- We will match terms in the exact optimality condition and the simulation-based version.
- Let $\hat{\xi}_i^t$ be the relative frequency of i in row sampling up to time t.
- We have

$$\frac{1}{t+1}\sum_{k=0}^{t}\phi(i_k)\phi(i_k)' = \sum_{i=1}^{n}\hat{\xi}_i^t\phi(i)\phi(i)' \approx \sum_{i=1}^{n}\xi_i\phi(i)\phi(i)'$$

$$\frac{1}{t+1}\sum_{k=0}^{t}\phi(i_k)b_{i_k} = \sum_{i=1}^{n}\hat{\xi}_i^t\phi(i)b_i \approx \sum_{i=1}^{n}\xi_i\phi(i)b_i$$

- Let $\hat{p}_{ij}^t$ be the relative frequency of (i,j) in column sampling up to time t.

$$\begin{aligned}\frac{1}{t+1}\sum_{k=0}^{t}\frac{a_{i_kj_k}}{p_{i_kj_k}}\phi(i_k)\phi(j_k)' &\\ &= \sum_{i=1}^{n}\hat{\xi}_i^t\sum_{j=1}^{n}\hat{p}_{ij}^t\frac{a_{ij}}{p_{ij}}\phi(i)\phi(j)' \\ &\approx \sum_{i=1}^{n}\xi_i\sum_{j=1}^{n}a_{ij}\phi(i)\phi(j)'\end{aligned}$$

6.231 DYNAMIC PROGRAMMING

LECTURE 24

LECTURE OUTLINE

- Additional topics in ADP
- Nonlinear versions of the projected equation
- Extension of Q-learning for optimal stopping
- Basis function adaptation
- Gradient-based approximation in policy space

NONLINEAR EXTENSIONS OF PROJECTED EQ

- If the mapping T is nonlinear (as for example in the case of multiple policies) the projected equation $\Phi r = \Pi T(\Phi r)$ is also nonlinear.
- Any solution r^* satisfies

$$r^* \in \arg\min_{r\in\Re^s} \left\|\Phi r - T(\Phi r^*)\right\|^2$$

or equivalently

$$\Phi'\big(\Phi r^* - T(\Phi r^*)\big) = 0$$

This is a nonlinear equation, which may have one or many solutions, or no solution at all.

- If ΠT is a contraction, then there is a unique solution that can be obtained (in principle) by the fixed point iteration

$$\Phi r_{k+1} = \Pi T(\Phi r_k)$$

- We have seen a nonlinear special case of projected value iteration/LSPE where ΠT is a contraction, namely optimal stopping.
- This case can be generalized.

LSPE FOR OPTIMAL STOPPING EXTENDED

- Consider a system of the form

$$x = T(x) = Af(x) + b,$$

where $f : \Re^n \mapsto \Re^n$ is a mapping with scalar components of the form $f(x) = \big(f_1(x_1), \ldots, f_n(x_n)\big)$.

- Assume that each $f_i : \Re \mapsto \Re$ is nonexpansive:

$$\big|f_i(x_i) - f_i(\bar{x}_i)\big| \leq |x_i - \bar{x}_i|, \qquad \forall\ i,\ x_i,\ \bar{x}_i \in \Re$$

This guarantees that T is a contraction with respect to any weighted Euclidean norm $\|\cdot\|_\xi$ whenever A is a contraction with respect to that norm.

- Algorithms similar to LSPE [approximating $\Phi r_{k+1} = \Pi T(\Phi r_k)$] are then possible.

- Special case: In the optimal stopping problem of Section 6.4, x is the Q-factor corresponding to the continuation action, $\alpha \in (0,1)$ is a discount factor, $f_i(x_i) = \min\{c_i, x_i\}$, and $A = \alpha P$, where P is the transition matrix for continuing.

- If $\sum_{j=1}^n p_{\bar{i}j} < 1$ for some state $\bar{i}$, and $0 \leq P \leq Q$, where Q is an irreducible transition matrix, then $\Pi\big((1-\gamma)I + \gamma T\big)$ is a contraction with respect to $\|\cdot\|_\xi$ for all $\gamma \in (0,1)$, even with $\alpha = 1$.

BASIS FUNCTION ADAPTATION I

- An important issue in ADP is how to select basis functions.

- A possible approach is to introduce basis functions that are parametrized by a vector θ, and optimize over θ, i.e., solve the problem

$$\min_{\theta \in \Theta} F\big(\tilde{J}(\theta)\big)$$

where $\tilde{J}(\theta)$ is the solution of the projected equation.

- One example is

$$F\big(\tilde{J}(\theta)\big) = \big\|\tilde{J}(\theta) - T\big(\tilde{J}(\theta)\big)\big\|^2$$

- Another example is

$$F\big(\tilde{J}(\theta)\big) = \sum_{i \in I} |J(i) - \tilde{J}(\theta)(i)|^2,$$

where I is a subset of states, and $J(i)$, $i \in I$, are the costs of the policy at these states calculated directly by simulation.

BASIS FUNCTION ADAPTATION II

- Some algorithm may be used to minimize $F\big(\tilde{J}(\theta)\big)$ over θ.
- A challenge here is that the algorithm should use low-dimensional calculations.
- One possibility is to use a form of random search (the cross-entropy method); see the paper by Menache, Mannor, and Shimkin (Annals of Oper. Res., Vol. 134, 2005)
- Another possibility is to use a gradient method. For this it is necessary to estimate the partial derivatives of $\tilde{J}(\theta)$ with respect to the components of θ.
- It turns out that by differentiating the projected equation, these partial derivatives can be calculated using low-dimensional operations. See the paper by Menache, Mannor, and Shimkin, and a recent paper by Yu and Bertsekas (2009).

APPROXIMATION IN POLICY SPACE I

- Consider an average cost problem, where the problem data are parametrized by a vector r, i.e., a cost vector $g(r)$, transition probability matrix $P(r)$. Let $\eta(r)$ be the (scalar) average cost per stage, satisfying Bellman's equation

$$\eta(r)e + h(r) = g(r) + P(r)h(r)$$

where $h(r)$ is the corresponding differential cost vector.

- Consider minimizing $\eta(r)$ over r (here the data dependence on control is encoded in the parametrization). We can try to solve the problem by nonlinear programming/gradient descent methods.

- **Important fact:** If $\Delta\eta$ is the change in η due to a small change Δr from a given r, we have

$$\Delta\eta = \xi'(\Delta g + \Delta P h),$$

where ξ is the steady-state probability distribution/vector corresponding to $P(r)$, and all the quantities above are evaluated at r:

$$\Delta\eta = \eta(r + \Delta r) - \eta(r),$$

$$\Delta g = g(r+\Delta r)-g(r), \qquad \Delta P = P(r+\Delta r)-P(r)$$

APPROXIMATION IN POLICY SPACE II

- **Proof of the gradient formula:** We have, by "differentiating" Bellman's equation,

$$\Delta\eta(r)\cdot e + \Delta h(r) = \Delta g(r) + \Delta P(r)h(r) + P(r)\Delta h(r)$$

By left-multiplying with ξ',

$$\xi'\Delta\eta(r)\cdot e + \xi'\Delta h(r) = \xi'\big(\Delta g(r) + \Delta P(r)h(r)\big) + \xi' P(r)\Delta h(r)$$

Since $\xi'\Delta\eta(r)\cdot e = \Delta\eta(r)$ and $\xi' = \xi' P(r)$, this equation simplifies to

$$\Delta\eta = \xi'(\Delta g + \Delta P h)$$

- Since we don't know ξ, we cannot implement a gradient-like method for minimizing $\eta(r)$. An alternative is to use "sampled gradients", i.e., generate a simulation trajectory $(i_0, i_1, \ldots)$, and change r once in a while, in the direction of a simulation-based estimate of $\xi'(\Delta g + \Delta P h)$.
- 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.

6.231 DYNAMIC PROGRAMMING

OVERVIEW-EPILOGUE

LECTURE OUTLINE

- Finite horizon problems
  - Deterministic vs Stochastic
  - Perfect vs Imperfect State Info
- Infinite horizon problems
  - Stochastic shortest path problems
  - Discounted problems
  - Average cost problems

FINITE HORIZON PROBLEMS - ANALYSIS

- Perfect state info
  - A general formulation - Basic problem, DP algorithm
  - A few nice problems admit analytical solution
- Imperfect state info
  - Sufficient statistics - Reduction to perfect state info
  - Very few nice problems that admit analytical solution
  - Finite-state problems admit reformulation as perfect state info problems whose states are prob. distributions (the belief vectors)

FINITE HORIZON PROBS - EXACT COMP. SOL.

- Deterministic finite-state problems
  - Equivalent to shortest path
  - A wealth of fast algorithms
  - Hard combinatorial problems are a special case (but # of states grows exponentially)
- Stochastic perfect state info problems
  - The DP algorithm is the only choice
  - Curse of dimensionality is big bottleneck
- Imperfect state info problems
  - Forget it!
  - Only trivial examples admit an exact computational solution

FINITE HORIZON PROBS - APPROX. SOL.

- Many techniques (and combinations thereof) to choose from
- Simplification approaches
  - Certainty equivalence
  - Problem simplification
  - Rolling horizon
  - Aggregation - Coarse grid discretization
- Limited lookahead combined with:
  - Rollout
  - MPC (an important special case)
  - Feature-based cost function approximation
- Approximation in policy space
  - Gradient methods
  - Random search

INFINITE HORIZON PROBLEMS - ANALYSIS

- A more extensive theory
- Bellman's equation
- Optimality conditions
- Contraction mappings
- A few nice problems admit analytical solution
- Idiosynchracies of average cost problems
- Idiosynchracies of problems with no underlying contraction
- Elegant analysis

INF. HORIZON PROBS - EXACT COMP. SOL.

- Value iteration
  - Variations (asynchronous, modified, etc)
- Policy iteration
  - Variations (asynchronous, based on value iteration, etc)
- Linear programming
- Elegant algorithmic analysis
- Curse of dimensionality is major bottleneck

INFINITE HORIZON PROBS - ADP

- Approximation in value space (over a subspace of basis functions)
- Approximate policy evaluation
  - Direct methods
  - Indirect methods (projected equation methods)
  - Aggregation methods - Coarse grid discretization
- Q-Learning
  - Exact Q-factor computation by simulation
  - Approximate Q-factor computation by simulation
  - Projected equation methods for optimal stopping
  - Aggregation-based Q-learning
- Approximate LP
- Approximation in policy space
  - Gradient methods
  - Random search