6.231 DYNAMIC PROGRAMMING

LECTURE 19

LECTURE OUTLINE

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

AVERAGE COST PER STAGE PROBLEM

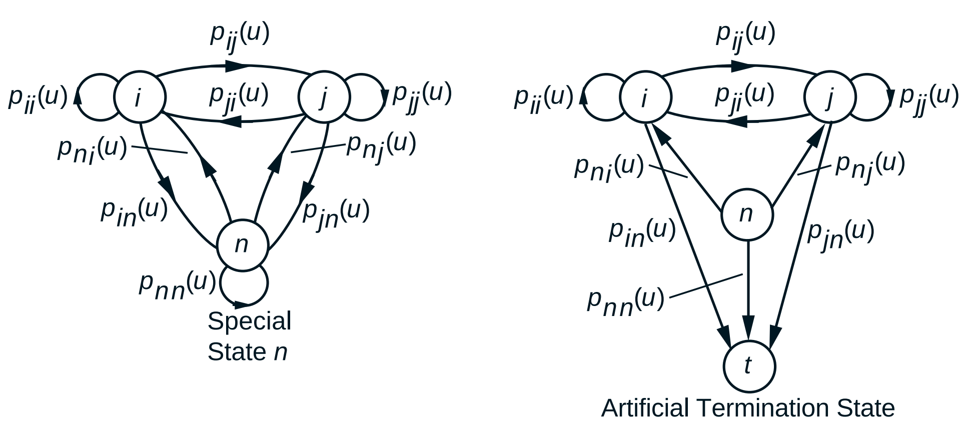
- Stationary system with finite number of states and controls
- Minimize over policies $\pi = \{\mu_0, \mu_1, ...\}$

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

- 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 $J^*(i) \leq J^*(j)$]. So $J^*(i) \equiv \lambda^*$ for all i.
  - Because "communication" issues are so important, the methodology relies heavily on Markov chain theory.

CONNECTION WITH SSP

- Assumption: State n is such that for some integer $m > 0$, and for all initial states and all policies, n is visited with positive probability at least once within the first m stages.
- Divide the sequence of generated states into cycles marked by successive visits to n.
- Each of the cycles can be viewed as a state trajectory of a corresponding stochastic shortest path problem with n as the termination state.



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

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

CONNECTION WITH SSP (CONTINUED)

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

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

where for a fixed μ,

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

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

- Intuitively, optimal cycle cost $= \lambda^*$, so

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

with equality if μ is optimal.

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

BELLMAN'S EQUATION

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

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

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

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

- Combining these equations, we have

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

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

MORE ON THE CONNECTION WITH SSP

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

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

- 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: Find for all i

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

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

- Result: For each k, we either have $\lambda^{k+1} < \lambda^k$ or

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

The algorithm terminates with an optimal policy.