6.231 DYNAMIC PROGRAMMING

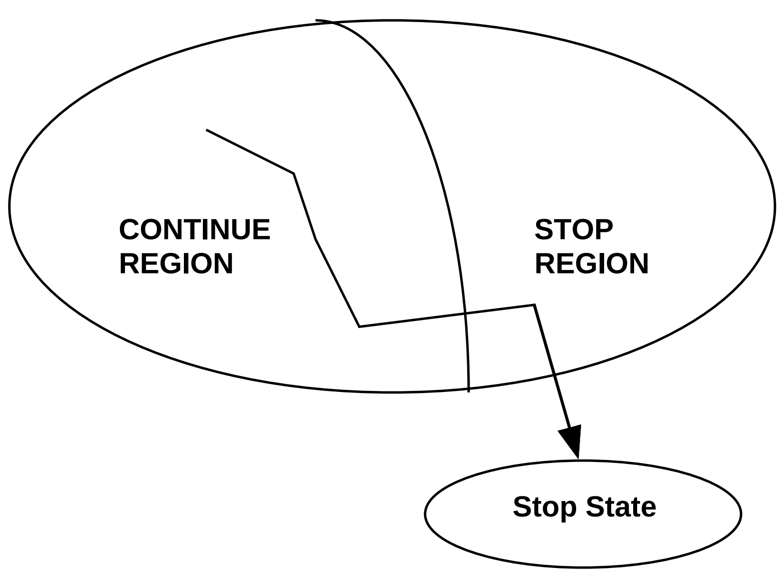
LECTURE 6

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;

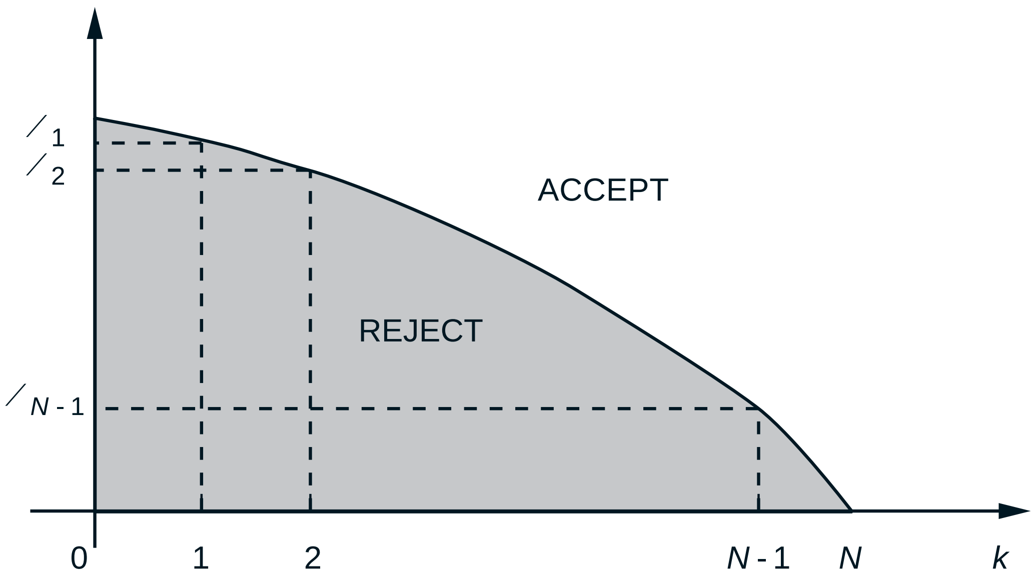
$$\text{accept the offer } x_k \qquad \text{if } x_k > \alpha_k,$$

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

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

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

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

Consider the list with i and j interchanged

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

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

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

MINIMAX CONTROL

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

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

- The DP algorithm takes the form

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

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

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

UNKNOWN-BUT-BOUNDED CONTROL

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

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

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

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

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

- We must reach at time k the set

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

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

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

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