6.231 Dynamic Programming

Midterm, Fall 2008

Instructions

The midterm comprises three problems. Problem 1 is worth 60 points, problem 2 is worth 40 points, and problem 3 is worth 40 points. Your grade $G \in [0,\ 100]$ is given by the following formula:

$$G = 60 \cdot f_1 + 40 \cdot \min[1,\ f_2 + f_3],$$

where $f_i \in [0,\ 1]$, $i = 1, 2, 3$, is the fraction of problem i that you solved correctly.

Notice that if you solve correctly both problems 2 and 3, but you do not solve problem 1, your grade will only be $G = 40$. Thus, try to solve problem 1 first, then choose between problems 2 and 3 (whichever you prefer), and if you still have time, try to do the remaining problem.

Credit is given only for **clear** logic and **careful** reasoning.

GOOD LUCK!

Problem 1 (Infinite Horizon Problem, 60 points)

An engineer has invented a better mouse trap and is interested in selling it for the right price. At the beginning of each period, he receives a sale offer that takes one of the values $s_1, \ldots, s_n$ with corresponding probabilities $p_1, \ldots, p_n$, independently of prior offers. If he accepts the offer he retires from engineering. If he refuses the offer, he may accept subsequent offers but he also runs the risk that a competitor will invent an even better mouse trap, rendering his own unsaleable; this happens with probability $\beta > 0$ at each time period, independently of earlier time periods. While he is overtaken by the competitor, at each time period, he may choose to retire from engineering, or he may choose to invest an amount $v \geq 0$, in which case he has a probability γ to improve his mouse trap, overtake his competitor, and start receiving offers as earlier. The problem is to determine the engineer's strategy to maximize his discounted expected payoff (minus investment cost), assuming a discount factor $\alpha < 1$.

(a) [15 points] Formulate the problem as an infinite horizon discounted cost problem and write the corresponding Bellman's equation.

(b) [15 points] Characterize as best as you can an optimal policy.

(c) [10 points] Describe as best as you can how policy iteration would work for this problem. What kind of initial policy would lead to an efficient policy iteration algorithm?

(d) [10 points] Assume that there is no discount factor. Does the problem make sense as an average cost per stage problem?

(e) [10 points] Assume that there is no discount factor and that the investment cost v is equal to 0. Does the problem make sense as a stochastic shortest path problem, and what is then the optimal policy?

Solution to Problem 1

(a) Let the set of possible offers be $\{x_i | i = 1, \ldots, n\} \cup \{t, r\}$, where being in a state has the following meaning:

$$x_i\text{: receiving offer } s_i; \qquad t\text{: overtaken}; \qquad r\text{: retired.}$$

Define the control space by $\{A(\text{accept offer}), I(\text{invest}), R(\text{retire}), Rej(\text{reject})\}$. The state r is absorbing, and for the other states, the set of admissible controls is

$$U(x_i) = \{A, Rej\}, \qquad U(t) = \{I, R\}.$$

Define the corresponding transition probabilities, and per-stage costs as described in the problem. Bellman's equation for $\alpha < 1$ is

$$J^*(x_i) = \max \left[s_i, \alpha \left((1-\beta) \sum_{j=1}^{n} p_j J^*(x_j) + \beta J^*(t) \right) \right], \tag{1}$$

$$J^*(t) = \max \left[0, -v + \alpha \left(\gamma \sum_{j=1}^{n} p_j J^*(x_j) + (1-\gamma) J^*(t) \right) \right]. \tag{2}$$

(b) From Eq. (1), we see that since the second term in the minimization of the right-hand side does not depend on the current state x_i, the optimal policy is a threshold policy at states x_i, $i = 1, \ldots, n$. A single threshold is needed since $J^*(s)$ is clearly monotonically nondecreasing with s. From Eq. (2), we see that once the inventor has been overtaken, it is either optimal to retire immediately, or to keep investing until his mouse trap is improved.

(c) Consider policy iteration. If the current policy μ is a threshold policy, the cost $J_\mu(s)$ is monotonically nondecreasing with s, so the cost of the improved policy is also monotonically nondecreasing with s. Hence, starting with a nonoptimal threshold policy, the method will generate a new and improved policy (for at least one offer). It follows that starting with a threshold policy, after at most n iterations, policy iteration will terminate with an optimal threshold policy.

(d) Even without discounting, all policies have finite total cost, except the one that never sells and always invests, which incurs infinite total cost. Thus, we see that under the average cost criterion, all of the finite total cost policies will be optimal. Thus, the average cost criterion does not discriminate enough between different policies and makes no sense.

(e) Since there is no discounting and $v = 0$, there is no penalty for waiting as long as necessary until the maximum possible offer is received, which is going to happen with probability 1. So a stochastic shortest path formulation makes limited sense, since it excludes from consideration all offers except the maximum possible, no matter how unlikely this maximum offer is.

Problem 2 (Imperfect State Information Problem, 40 points)

A machine tosses a coin N times, but each time may switch the probability of heads between two possible values h and $\overline{h}$. The switch occurs with probability q at each time, independently of the switches in previous times. A gambler may bet on the outcome of each coin toss knowing h, $\overline{h}$, and q, the outcomes of previous tosses, and the probability r that h is used at the initial toss. If the gambler bets on a coin toss, he wins \$2 if a head occurs, while he loses \$1 if a tail occurs. The gambler may also (permanently) stop betting prior to any coin toss.

(a) [20 points] Define

$$p_k = \begin{cases} r & \text{for } k = 0, \\ Prob\{\text{Coin with a probability of heads } h \text{ is used at toss } k \,|\, z_{k-1}, \ldots, z_0\} & \text{for } k = 1, \ldots, N-1, \end{cases}$$

where $z_{k-1}, \ldots, z_0$ are the preceding observed tosses. Write a equation for the evolution of p_k as the coin tosses are observed.

(b) [10 points] Use the equation of part (a) to write a DP algorithm to find the gambler's optimal policy for betting or stopping to bet prior to any coin toss. (You will receive credit if you do this part correctly, assuming a generic functional form of this equation.)

(c) [10 points] Repeat part (b) for the variant of the problem where the gambler may resume play at any time after he stops betting, and he keeps observing the coin tosses also when he is not betting.

Solution to Problem 2

(a) The machine uses two coins, one with a probability of heads equal to h, and the the other one with a probability of heads equal to $\overline{h}$. We call the first coin a type h coin, while we call the second coin a type $\overline{h}$ coin.

Let $f_h(z)$, $z \in \{head, tail\}$, be the probability mass function for the type h coin (clearly, $f_h(head) = h$). Similarly, let $f_{\overline{h}}(z)$, $z \in \{head, tail\}$, be the probability mass function for the type $\overline{h}$ coin (clearly, $f_{\overline{h}}(head) = \overline{h}$).

We are faced with an imperfect state information problem where the state at stage k, that we call x_k, is the type of coin that the machine uses at stage k. We write $x_k = h$ if at stage k the machine uses a type h coin, and similarly we write $x_k = \overline{h}$ if at stage k the machine uses a type $\overline{h}$ coin.

The conditional probability p_k is generated recursively according to the following equation (with $p_0 = r$):

$$\begin{aligned} p_{k+1} &= \\ &= Pr\{x_{k+1} = h \,|\, z_k, z_{k-1}, \ldots, z_0\} \\ &= \frac{Pr\{x_{k+1} = h, z_k \,|\, z_{k-1}, \ldots, z_0\}}{Pr\{z_k \,|\, z_{k-1}, \ldots, z_0\}} \\ &= \frac{Pr\{z_k \,|\, x_{k+1} = h, z_{k-1}, \ldots, z_0\}\, Pr\{x_{k+1} = h \,|\, z_{k-1}, \ldots, z_0\}}{Pr\{z_k \,|\, x_k = h, z_{k-1}, \ldots, z_0\} Pr\{x_k = h \,|\, z_{k-1}, \ldots, z_0\} + Pr\{z_k \,|\, x_k = \overline{h}, z_{k-1}, \ldots, z_0\} Pr\{x_k = \overline{h} \,|\, z_{k-1}, \ldots, z_0\}} \\ &= \frac{Pr\{z_k \,|\, x_{k+1} = h, z_{k-1}, \ldots, z_0\} \left(Pr\{x_{k+1} = h \,|\, x_k = h, z_{k-1}, \ldots, z_0\} p_k + Pr\{x_{k+1} = h \,|\, x_k = \overline{h}, z_{k-1}, \ldots, z_0\}(1 - p_k) \right)}{f_h(z_k) p_k + f_{\overline{h}}(z_k)(1 - p_k)} \end{aligned}$$

$$= \frac{\left[f_h(z_k)\left(\frac{(1-q)p_k}{(1-q)p_k+q(1-p_k)}\right) + f_{\overline{h}}(z_k)\left(\frac{q(1-p_k)}{(1-q)p_k+q(1-p_k)}\right)\right]\left((1-q)p_k + q(1-p_k)\right)}{f_h(z_k)p_k + f_{\overline{h}}(z_k)(1-p_k)}$$

$$\doteq \phi(p_k, z_k),$$

since

$$\begin{aligned}
&Pr\{z_k \,|\, x_{k+1} = h, z_{k-1}, \ldots, z_0\} = \\
&= Pr\{z_k \,|\, x_{k+1} = h, x_k = h, z_{k-1}, \ldots, z_0\}\, Pr\{x_k = h \,|\, x_{k+1} = h, z_{k-1}, \ldots, z_0\} + \\
&+ Pr\{z_k \,|\, x_{k+1} = h, x_k = \overline{h}, z_{k-1}, \ldots, z_0\}\, Pr\{x_k = \overline{h} \,|\, x_{k+1} = h, z_{k-1}, \ldots, z_0\} = \\
&= f_h(z_k)\left(\frac{Pr\{x_{k+1} = h \,|\, x_k = h, z_{k-1}, \ldots, z_0\}p_k}{Pr\{x_{k+1} = h \,|\, x_k = h, z_{k-1}, \ldots, z_0\}p_k + Pr\{x_{k+1} = h \,|\, x_k = \overline{h}, z_{k-1}, \ldots, z_0\}(1-p_k)}\right) + \\
&+ f_{\overline{h}}(z_k)\left(\frac{Pr\{x_{k+1} = h \,|\, x_k = \overline{h}, z_{k-1}, \ldots, z_0\}(1-p_k)}{Pr\{x_{k+1} = h \,|\, x_k = h, z_{k-1}, \ldots, z_0\}p_k + Pr\{x_{k+1} = h \,|\, x_k = \overline{h}, z_{k-1}, \ldots, z_0\}(1-p_k)}\right) = \\
&= f_h(z_k)\left(\frac{(1-q)p_k}{(1-q)p_k + q(1-p_k)}\right) + f_{\overline{h}}(z_k)\left(\frac{q(1-p_k)}{(1-q)p_k + q(1-p_k)}\right)
\end{aligned}$$

(b) The DP algorithm is

$$J_k(p_k) = \max\left[\underbrace{0}_{stop},\ 2\cdot(p_k h + (1-p_k)\overline{h}) - 1\cdot(p_k(1-h) + (1-p_k)(1-\overline{h})) + E_{z_k}\left\{J_{k+1}\big(\phi_k(p_k, z_k)\big)\right\}\right], \qquad k = 0, \ldots, N-1,$$

with $J_N(p_N) \equiv 0$.

(c) In this case we have for $k = 0, \ldots, N-1$

$$J_k(p_k) = \max\left[\underbrace{0 + E_{z_k}\left\{J_{k+1}\big(\phi_k(p_k, z_k)\big)\right\}}_{do\ not\ play},\ 2\cdot(p_k h + (1-p_k)\overline{h}) - 1\cdot(p_k(1-h) + (1-p_k)(1-\overline{h})) + E_{z_k}\left\{J_{k+1}\big(\phi_k(p_k, z_k)\big)\right\}\right],$$

with $J_N(p_N) \equiv 0$.

Thus, we do not play if

$$0 > 2\cdot(p_k h + (1-p_k)\overline{h}) - 1\cdot(p_k(1-h) + (1-p_k)(1-\overline{h})),$$

that is, assuming without loss of generality that $h > \overline{h}$, we do not play if

$$p_k < \frac{1 - 3\overline{h}}{3(h - \overline{h})}.$$

Problem 3 (Optimal Stopping Problem, 40 points)

A driver is looking for parking on the way to his destination. Each parking place is free with probability p independently of whether other parking places are free or not. The driver cannot observe whether a parking place is free until he reaches it. If he parks k places from his destination, he incurs a cost k. If he reaches the destination without having parked the cost is C. The driver wishes to find an optimal parking policy.

(a) [20 points] Formulate the problem as a DP problem, clearly identifying, states, controls, transition probabilities and costs.

(b) [20 points] Write the DP algorithm and show that it can be represented by the sequence $\{C_k\}$ generated as follows

$$C_k = p\min[k, C_{k-1}] + qC_{k-1}, \qquad k = 1, 2, \ldots,$$

where $C_0 = C$ and $q = 1 - p$. Characterize as best as you can the optimal policy.

Solution to Problem 3

(a) Let the state $x_k \in \{T, \bar{T}\}$ where T represents the driver having parked before reaching the kth spot. Let the control at each parking spot $u_k \in \{P, \bar{P}\}$ where P represents the choice to park in the kth spot. Let the disturbance w_k equal 1 if the kth spot is free; otherwise it equals 0. Clearly, we have the control constraint that $u_k = \bar{P}$, if $x_k = T$ or $w_k = 0$. The cost associated with parking in the kth spot is:

$$g_k(\bar{T}, P, 1) = k$$

If the driver has not parked upon reaching his destination, he incurs a cost $g_N(\bar{T}) = C$. All other costs are zero. The system evolves according to:

$$x_{k+1} = \begin{cases} T, & \text{if } x_k = T \text{ or } u_k = P \\ \bar{T}, & \text{otherwise} \end{cases}$$

(b) Once the driver has parked, his remaining cost is zero. Thus, we can define C_k to be the expected remaining cost, given that the driver has not parked before the kth spot. (Note that this is simply $J_k(\bar{T})$). The DP algorithm starts with

$$C_0 = C,$$

and for $k = 1, 2, \ldots$, generates C_k by

$$C_k = \min_{u_k \in \{P, \bar{P}\}} \mathop{E}_{w_k}\big\{g_k(\bar{T}, u_k, w_k) + J_{k-1}(x_{k-1})\big\} = \min\Big[\underbrace{p\big[k + J_{k-1}(T)\big]}_{park,\ free} + \underbrace{qJ_{k-1}(\bar{T})}_{park,\ not\ free}\ ,\quad \underbrace{J_{k-1}(\bar{T})}_{don't\ park}\Big]$$

Since $J_k(T) = 0$ for all k, we have

$$C_k = \min\big[pk + qC_{k-1}, C_{k-1}\big] = p\min\big[k, C_{k-1}\big] + qC_{k-1}.$$

Thus the sequence $\{C_k\}$ is the sequence generated by DP.

6.231 Dynamic Programming and Stochastic Control
Fall 2011