Stochastic Modeling and Optimization Problemset 1

Aimee Barciauskas, Andreas Lloyd, Francis Thomas Moynihan IV, and Seda Yilmaz

26 February 2016

1. Machine maintenance

- x_k - (*state*) Machine is working or broken
- u_k - (*control*) Whether to perform maintenance
- w_k - (*uncertainty*) Machine breaks or doesn't break

There are no additional dynamics or constraints for this problem.

Expected profit is always $\$100 - g(u)$ where $g(u)$ is the exepected cost of decision u, (i.e.: $\mathbb{E}_w\{G(u, w)\}$). By minimizing expected cost we maximize expected profit.

At the beginning of each week, we much pick an optimal strategy given x_k: the machine is working or broken.

State 1: when the machine is working, (the initial state), we have two strategies and their associated expected costs:

Option 1: Perform Maintenence

If maintenance is performed, the expected cost is 20 (the cost of maintenance) plus the probability the machine breaks during the week times lost profit:

$J_{\mu_0} = 20 + 0.4 * 100 = 60$

Option 2: Do nothing

If maintenance is not performed, the expected cost is the probability the machine breaks down times lost profit.

$J_{\mu_1} = 0.7 * 100 = 70$

Strategy when the machine is working, performing maintenance minimizes expected cost.

State 2: when the machine is broken, there are three strategies:

Option 1: Buy a new machine.

$J_{\mu_0} = 90$

Option 2: Repair the machine.

$J_{\mu_1} = 40 + 0.4 * 100 = 80$

Option 3: Do nothing.

$J_{\mu_2} = 100$

When the machine is broken, making a repair minimizes expected cost

The optimal strategy which minimizes expected cost (and maximizes expected profit) is to perform preventative maintenance when the machine is working and repair the machine when it is broken.

Alternately, we can state these strategies as the policies:

$\pi = \mu_k(x_k) = u_k$

$\mu_k(working) =$ perform preventative maintenance

$\mu_k(broken) =$ repair broken machine

DP Algorithm

1. $J^*(x_0) = min_\pi J_\pi(x_0) = \mu_k(working)$
2. From week $1, .., 4$, act according to policies $\pi = \mu(broken), \mu(working)$, as the policies are detailed above.

2. Discounted Cost

In the framework of the basic problem, consider the case where the cost is of the form:

$$\mathbb{E}_{w_k}\Bigg\{\alpha^N g_N(x_N) + \sum_{k=0}^{N-1} \alpha^k g_k(x_k, u_k, w_k)\Bigg\}$$

where $\alpha \in (0,1)$ is a discount factor. Develop a DP-like algorithm for this problem.

$$J_N(x_N) = \alpha^N g_N(x_N) <=> J_N(x_N)\alpha^{-N} = g_N(x_N)$$

$$J_k(x_k) = \min_{u_k} \mathbb{E}_{w_k}\Bigg\{\alpha^k g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)\Bigg\}$$

$$J_k(x_k)\alpha^{-k} = \min_{u_k} \mathbb{E}_{w_k}\Bigg\{g_k(x_k, u_k, w_k) + \alpha^{-k} J_{k+1}(f_k(x_k, u_k, w_k)\Bigg\}$$

$$J_k(x_k)\alpha^{-k} = \min_{u_k} \mathbb{E}_{w_k}\Bigg\{g_k(x_k, u_k, w_k) + \alpha\alpha^{-k+1} J_{k+1}(f_k(x_k, u_k, w_k)\Bigg\}$$

Let $\phi_k := J_k x_k \alpha^{-k}$, then for the DP algorithm:

Initial step: $\phi_N(x_N) = g_N(x_N)$

Recursive function: $\phi_k(x_k) = \min_{u_k} \mathbb{E}_{w_k}\Bigg\{g_k(x_k, u_k, w_k) + \alpha\phi_{k+1}(f_k(x_k, u_k, w_k))\Bigg\}$

3. Multiplicative cost

In the framework of the basic problem, consider the case where the cost has the multiplicative form:

$$\mathbb{E}_{w_k}\Bigg\{g_N(x_N) g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) ... g_0(x_0, u_0, w_0)\Bigg\}$$

The primitives of the problem are the same as for the generic DP framework:

- x_k - state at time k
- u_k - control at time k (a strategy)
- w_k - uncertainty at time k, the realization of a random variable

The DP algorithm proceeds as follows:

Step 1. Start with:

$J_N(x_N) = g_N(x_N)$

Step 2. Work backwards in time

At time i, select the u_k which minimizes the expected cost:

$$J_k(x_k) = \min_{u_k \in U_k} \mathbb{E}_{w_k} \left\{ g_N(x_N) \prod_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

And the optimal policy becomes the $\mu_i^*, ..., \mu_{N-1}^*$ which result from this optimizes.

$$J_k^*(x_k) = J_k(x_k)$$

The policy which minimizes the expected cost is equivalent to a shortest path problem and thus satisfies the principal of optimality: the policies which optimize the tail subproblems are necessarily also a part of the optimal policy. Minimizing cost working backwards solves for the minimal cost.

4. Knapsack Problem

- *states:* z_i weight left after adding item i and previous items stored and amount
- *controls:* u_i adding a weight $x_i w_i$

$$u_i(x_i) \in \{0 \leq u_i \geq z_i, z_i \in [0, Z]\}$$

- *uncertainty:* None
- *dynamics:* $f_i(x_i, u_i) = z_{i+1} = z_i - x_i w_i$
- *cost:*
$$g_i(x_i, u_i) = \frac{(z_i - x_i w_i)}{w_i} v_i$$
  The cost of not filling up the remaining space with that item.

DP algorithm

$$g_N(x_N) = J_N(x_N) = \frac{(z_N - x_N w_N)}{w_N} v_N$$

Filling up the remaining space with the last item.

$$J_i(x_i) = \min_{u_i} \left\{ g_i(x_i, u_i) + J_{i+1}(f_i(x_i, u_i)) \right\}$$

5. Traveling Repairman Problem.

- *states:* x_k Sites served so far
- *controls:* u_k Site to serve next

$$u_k(x_k) \in \left\{s_{i-1}, s_{j+1}\right\}, x_k = \left\{s_i, s_{i+1}, ..., s_j\right\}$$

- *uncertainty:* w_k none
- *dynamics:* $f(x_k, u_k) = x_k + u_k = x_{k+1}$.
- *cost:*

$$g_k(x_k, u_k) = t_k + \sum^{U_k} c_i$$

where t_k is travelling cost from $u_{k-1}(x_{k-1})$ to $u_k(x_k)$ and c_i is the waiting cost of each site not yet visited. S is the set of all sites and $U_k = S - \left\{x_k\right\}$, e.g. we can chose from all sites not yet visited.

- *constraints:* Next site to serve must be adjacient to the sites served so far.

DP algorithm

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

$$J_k(x_k) = \min_{u_k} \left\{ g_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k)) \right\}$$