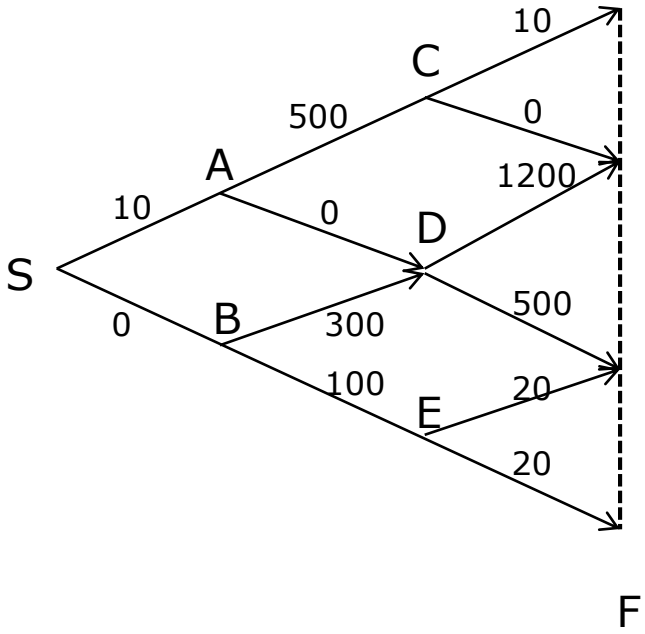
Last Time

Last time we took a forward approach and a backwards approach (policy evaluation algorithm (PEA)) and found the same results for the problem:



Question:

- Policy π: At vertex S go up. At all future *states* choose to go up if you have ever gone up in the past; otherwise choose to go down. (remember choosing to go up or down does not guarantee you will in this stochastic problem)
- What is the expected distance travelled under policy π ?

But the backward approach is more efficient.

Answer:
Forward approach: 601.12
PEA algorithm: 601.12

Theorem 4.2.1 guarantees these always give the same answer (under the conditions specified)

Today

Optimality equations for stochastic DPs:

- Why do they work?

Application to traversing a dangerous network

Principle of Optimality

Following is a quote from a 1957 paper by Richard Bellman:

"An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Optimization of Stochastic DPs

For policy π the expected discounted reward is:

$$u_N^{\pi}(s) = E_s^{\pi}[\sum_{t=1}^{N-1} \lambda^{t-1} r_t(X_t, Y_t) + r_N(X_N)]$$

Optimization Problem: An optimal policy π^* satisfies

$$u_N^{\pi^*}(s) \geq u_N^{\pi}(s), for\ all\ s \in S\ and\ for\ all\ polices\ \pi \in \Pi$$

To find π^* solve the following problem:

$$u_N^{\pi^*}(s) = \max_{\pi \in \Pi} u_N^{\pi}(s)$$

Principle of Optimality

Define the **optimal value to go** function as:

$$v_t(h_t) = \max_{\pi \in \Pi} \{v_t^{\pi}(h_t)\}$$

Obtained from policy evaluation algorithm

Optimality equations (**Bellman's equations**):

$$v_t(h_t) = \max_{a \in A} \{r_t(s_t, a) + \lambda \sum_{j \in S} p_t(j|s_t, a) v_{t+1}(h_{t+1})\}, t = 1, .., N-1, \ \forall h_t$$

$$v_N(h_N) = r_N(s_N), \forall h_N$$

Note: The optimality equations reduce to the policy evaluation equations when the "max" operator is replaced by a specific action

Principle of Optimality

Following is a fundamental result of dynamic programming

Theorem (≈4.3.2 Puterman): Suppose $\mathrm{v_t(h_t)}$, for all t and h_t is a solution to the optimality equations, then $v_t(h_t) = {v_t}^*(h_t)$, for all t and h_t and $v_1(s_1) = {u_N}^{\pi^*}(s_1)$, for all $s_1 \in S$.

Expected future rewards for all stages under the optimal policy π^*

Recall we are making the following assumptions:

- The set of actions, A, and states, S, are finite
- The rewards, r(s,a), are bounded, i.e., r(s,a) ≤ M
- The policy is deterministic (e.g. MD or HD)
- The decision maker's goal can be represented by linear additive rewards

Getting the Optimal Policy

The optimal policy is obtained using the optimal value function as follows…

$$d_t^*(h_t) \in argmax_{a \in A} \left\{ r_t(s_t, a) + \Sigma_{j \in S} p_t(j|s_t, a) v_{t+1}(h_{t+1}) \right\}$$

and the optimal policy is $\pi^* = (d_1^*, d_2^*, \ldots, d_{N-1}^*)$.

Optimality of HD and MD Policies

The following property can be used to establish optimality of policies constructed from deterministic decision rules:

Lemma (4.3.1 Puterman): Let $\mathrm{w}(\cdot)$ be some real valued function defined on discrete set W and let $\mathrm{q}(\cdot)$ be a probability distribution on W. Then

$$\max_{u \in W}\{w(u)\} \geq \sum_{u \in W} q(u)w(u)$$

Exercise: Given two policies, ${\pi_{HD}}^* \in \Pi^{HD}$ and ${\pi_{HR}}^* \in \Pi^{HR}$, that each satisfy their respective optimality equations, prove $v_t^{{\pi_{HD}}^*}(h_t) \geq v_t^{{\pi_{HR}}^*}(h_t), \forall\, h_t, t$

Optimality of MD Policies

Following is an aggregate of theorems in Puterman Section 4.4, based on the assumptions from last class, i.e., S and A are finite, bounded rewards, linear additive utility.

Theorem (≈4.2.2 Puterman): Let $\mathrm{v}_\mathrm{t}^*(\mathrm{h}_\mathrm{t}), \forall h_t$ be solutions of the optimality equations. If $\mathrm{v}_\mathrm{t}^*(\mathrm{h}_\mathrm{t})$ depends on h_t only through s_t then there is an MD optimal policy.

Proof: Completed in class.

Finding Optimal Policies

The following algorithm generalizes the policy evaluation algorithm to find an optimal policy when the optimal policy depends only on the current state.

Algorithm (Backward Induction):

1. Set $\mathrm{t} = \mathrm{N}, v_N^*(s_N) = r_N(s_N), \forall s_N$

2. *If* $t = 1$ *stop, otherwise go to step* 3

3. *Substitute* $t - 1$ *for* t *and compute* $v_t^*(s_t), \forall s_t,$ *as*:

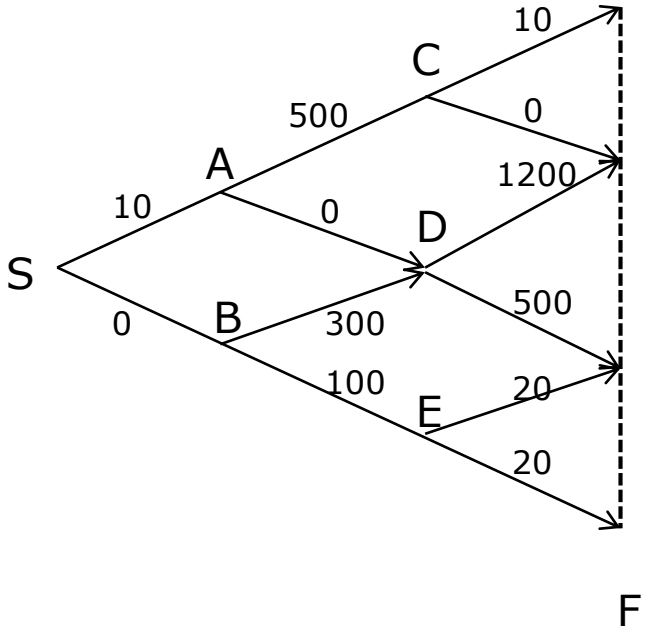
$$v_t^*(s_t) = \max_{\mathrm{a}\in A} \{\mathrm{r_t(s_t}, a) + \lambda \sum_{j\in S} p_t(j|s_t, a) v_{t+1}^*(j)\}$$

Set $a_t^*(s_t) = \arg\max_{\mathrm{a}\in A} \{\mathrm{r_t(s_t}, a) + \lambda \sum_{j\in S} p_t(j|s_t, a) v_{t+1}^*(j)\}$

return to step 2.

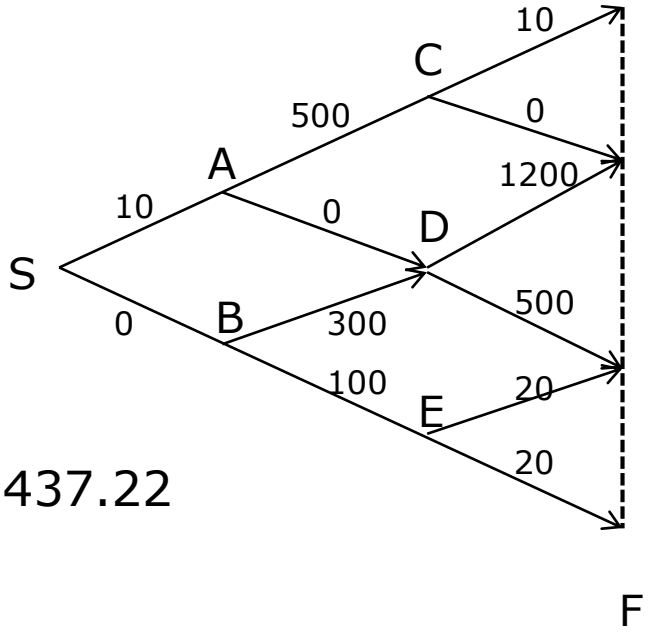
Example

- $\Pr(u|a=u) = 0.8$ and $\Pr(d|a=d) = 0.7$
- Use the backward induction algorithm to find the optimal policy that minimizes expected distance travelled.



Example

- $\Pr(u|a=u)=0.8$ and $\Pr(d|a=d)=0.7$
- Use the backward induction algorithm to find the optimal policy and minimum expected distance travelled.



Answer:

Expected distance travelled 437.22

Optimal Policy:

State: A B C D E

Action: u d d d u or d

Zombie Avoidance

Formulate and solve a DP for the following problem:

- You must traverse the following network in which edge weights are the probability of encountering a zombie if you travel along that edge
- Your goal is to minimize the probability of an encounter