Dynamic Programming

Operations Research

Anthony Papavasiliou

Contents

Table of Contents

Setting

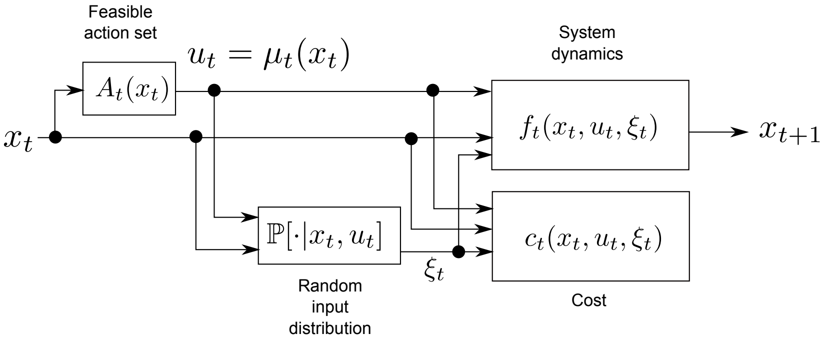
- **Dynamical** system with $H < \infty$ discrete time stages
  - Extensions exist for infinite horizon ($H = \infty$)
  - Extensions exist for continuous time
- **Controlled** system, denote u_t as *continuous* decision at stage t
- **Stochastic** system, denote ξ_t as *discrete* random vector at stage t
  - Extensions exist for continuous uncertainty
- Denote x_t as *continuous* **state** of the system at the *end* of stage t
  - State encodes everything we need to know, except ξ_t and u_t, for describing the evolution of the system
- Transition equation:

$$x_{t+1} = f_t(x_t, u_t, \xi_t)$$

Setting (II)

- *Markovian* uncertainty: we can define probability distribution $\mathbb{P}[\cdot|x_t, u_t]$ for ξ_t, *independently* of $\xi_{t-1}, \xi_{t-2}, \ldots, \xi_0$
- Denote $A_t(x_t)$ as set of finite actions at stage t
- Costs are *additive*, denote $c_t(x_t, u_t, \xi_t)$ as cost per time stage
- Usually (but not always), we will assume that x_t, ξ_t, and u_t live in $\mathbb{R}^n$

Block Diagram Representation



- The flow of information is consistent (everything depends on information that is already revealed)
- The process is repeated identically over stages

Sequence of Events

In a given time stage,

1. observe state x_t
2. decide u_t *after* observing x_t
3. sample ξ_t from a distribution that depends on x_t, u_t
4. Incur cost $c_t(x_t, u_t, \xi_t)$
5. Move to new state x_{t+1}

Policy

This is **not** your 'usual' optimization

- In 'usual' optimization we are looking for an optimal *vector* $x^\star$
- In multi-stage optimization under uncertainty we are looking for a *sequence of functions* $\mu_t(x_t)$
- The functions $\mu_t(x_t)$ are called a **policy**, they tell us what to do if we observe x_t in stage t

Objective

- Recall costs are additive
  - For $t = 0, \dots, H-1$, we incur cost $c_t(x_t, u_t, \xi_t)$
  - Assume final-period cost only depends on x_H, i.e. $c_H(x_H)$
- **Key observation**: *given* a policy $\mu_t(x_t)$, we can define a distribution for the sequence $(x_t, \xi_t), t = 0, \dots, H$
- Given a distribution for the sequence $(x_t, \xi_t), t = 0, \dots, H$, we can define expected cost

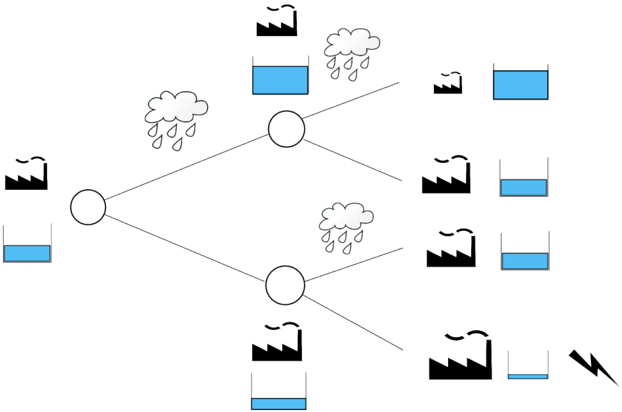
$$\mathbb{E}[\sum_{t=0}^{H-1} c_t(x_t, \mu_t(x_t), \xi_t) + c_H(x_H)]$$

Formal Problem Statement

We are looking for the **optimal policy**: the *policy* which minimizes *expected* cost

$$
\begin{aligned}
(MP): \quad & \min_{\mu_t} \mathbb{E}[\sum_{t=0}^{H-1} c_t(x_t, \mu_t(x_t), \xi_t) + c_H(x_H)] \\
& \mu_t(x_t) \in A_t(x_t) \\
& x_{t+1} = f_t(x_t, \mu_t(x_t), \xi_t)
\end{aligned}
$$

Recalling Hydrothermal Scheduling



- Too much water in dams leads to water spillage and unnecessary thermal generation costs
- Too little water in dams leads to load curtailment

Hydrothermal Scheduling Problem Statement

- Time-varying electricity demand D_t
- Three options
  - Hydro units: produce q_t at zero cost
  - Thermal units: produce p_t at marginal cost C
  - Load shedding: cut supply by l_t at marginal cost V
- Rainfall uncertainty: independent identical normal distribution with mean μ and standard deviation σ
- Hydro reservoir can store up to E units of energy
- Thermal generators can produce up to P units of power per period

Hydrothermal Scheduling Model Description

- *Continuous* action vector: $u_t = (p_t, q_t, l_t) \in \mathbb{R}^3$
- *Continuous* state vector $x_t \in \mathbb{R}$: level of reservoir at the *beginning* of stage t
- Feasible action set:

$$
\begin{aligned}
A_t(x_t) = \{(p_t, q_t, l_t) :& \\
q_t \leq x_t& \\
p_t + q_t + l_t = D_t& \\
p_t \leq P& \\
p_t, q_t, l_t \geq 0\}&
\end{aligned}
$$

- *Continuous* random disturbance $\xi_t \in \mathbb{R}$: rainfall

Hydrothermal Scheduling Model Description (II)

- Transition probability function:

$$\mathbb{P}[\xi_t \leq R] = \Phi(\frac{R - \mu}{\sigma}),$$

  where $\Phi(\cdot)$ is the cdf of a standard normal random variable

- System transition function:

$$x_{t+1} = f_t(x_t, u_t, \xi_t) = \min(E, x_t + \xi_t - q_t)$$

- Cost function:

$$c_t(x_t, u_t, \xi_t) = C \cdot p_t + V \cdot l_t$$

Hydrothermal Scheduling with AR Rainfall

Same problem as before, except rainfall r_t follows an autoregressive (AR) process:

$$r_t = c + \phi \cdot r_{t-1} + w_t$$

- c and ϕ are fixed parameters
- w_t: independent identical distribution according to a normal distribution with mean μ and standard deviation σ

Hydrothermal Scheduling with AR Rainfall: Model Formulation

- Redefine random disturbance as $\xi_t = w_t \in \mathbb{R}$
- State of the system: $x_t = (e_t, r_t)^T \in \mathbb{R}^2$
  - e_t: level of energy stored in the reservoir
  - r_t: rainfall
- System dynamic function:

$$x_{t+1} = (e_{t+1}, r_{t+1})^T = f_t(x_t, u_t, \xi_t) = \begin{bmatrix} \min(E, x_t + \xi_t - q_t) \\ c + \phi \cdot r_t + \xi_t \end{bmatrix}$$

Capacity Expansion Problem

- *Continuous* action vector $u_t = (z_t, y_t) \in \mathbb{R}^{nm+n-1}$
  - Amount of capacity constructed: $z_{it} = (z_{1t}, \ldots, z_{n-1,t}) \in \mathbb{R}^{n-1}$
  - Amount of power that from technology i to block j: $y_t = (y_{11t}, \ldots, y_{1mt}, \ldots, y_{n1t}, \ldots, y_{nmt}) \in \mathbb{R}^{nm}$
- *Continuous* state vector: capacity that has been constructed so far for each technology, $x_t = v_t = (v_{1t}, \ldots, v_{n-1,t}) \in \mathbb{R}^{n-1}$.
- *Discrete* uncertain demand $D_t = (D_{1t}, \ldots D_{mt}) \in \mathbb{R}^m$ with distribution $\mathbb{P}[\cdot]$, independent of x_t and u_t

Capacity Expansion Problem (II)

- Feasible action set:

$$
\begin{aligned}
& A_t(x_t) = \{(z_t, y_t) : \\
& \sum_{j=1}^{m} y_{ijt} \leq x_{it}, i = 1, \ldots, n-1, \\
& \sum_{i=1}^{n} y_{ijt} = D_j, j = 1, \ldots, m \\
& y_t \geq 0, z_t \geq 0\}
\end{aligned}
$$

- System transition function:

$$
x_{i,t+1} = x_{it} + z_{it}, i = 1, \ldots, n-1
$$

Capacity Expansion Problem (III)

- Cost function:

$$c_t(x_t, u_t, \xi_t) = \sum_{i=1}^{n-1} I_i \cdot z_{it} + \sum_{i=1}^{n} \sum_{j=1}^{m} C_i \cdot T_j \cdot y_{ijt},$$

  where
  - I_i: investment cost of technology i
  - C_i: marginal cost of technology i
  - T_j: (deterministic) duration of block j
- Note: capacity built in period t cannot be used for satisfying the demand of period t

Machine Scheduling: Problem Statement

- Machine produces P units of output when on
- Cost of C is paid for every period that the machine is on
- Machine output earns time-varying price λ_t
- Machine needs to stay on for at least 3 hours once started up

Machine Scheduling: Model Description

- Action set: $\mathbb{B} = \{\text{Stay}, \text{Change}\}$
- State: number of hours that have elapsed since the machine was last turned on, belongs to set $\mathbb{Z} = \{0, 1, 2, \ldots\}$ - 0 belongs to 'Off'
- Feasible action set:

$$
\begin{aligned}
A_t(0) &= \{\text{Stay}, \text{Change}\}, \\
A_t(x_t) &= \{\text{Stay}\}, x_t = 1, 2 \\
A_t(x_t) &= \{\text{Stay}, \text{Change}\}, x_t \geq 3
\end{aligned}
$$

Machine Scheduling: Model Description (II)

- System transition function:

$$\begin{aligned} x_{t+1} &= f_t(0, \text{Stay}) = 0 \\ x_{t+1} &= f_t(x_t, \text{Stay}) = x_t + 1, x_t \geq 1 \\ x_{t+1} &= f_t(0, \text{Change}) = 1 \\ x_{t+1} &= f_t(x_t, \text{Change}) = 0, x_t \geq 1 \end{aligned}$$

- Cost function:

$$\begin{aligned} c_t(x_t, u_t) &= (C - \lambda_t \cdot P), x_t \geq 1 \\ c_t(0, u_t) &= 0 \end{aligned}$$

Table of Contents

Value Function

- Solving (*MP*) means solving for a policy / mapping μ_t, not a vector u_t
- **Value function** $V_t(x_t)$: least expected cost if optimal decisions would be made from stage t onwards *given* state x_t

The Dynamic Programming Algorithm

Dynamic programming algorithm:

- Starting from $t = H$, for all $x_t \in A_t(x_t)$, compute

$$V_H(x_H) = c_H(x_H).$$

- Moving backwards in time, for all $t = H - 1, \ldots, 0$, for all $x_t \in A_t(x_t)$, compute

$$V_t(x_t) = \min_{u_t \in A_t(x_t)} \mathbb{E}_{\xi_t}[(c_t(x_t, u_t, \xi_t) + V_{t+1}(f_{t+1}(x_t, u_t, \xi_t)))|x_t, u_t]$$

  where the expectation is over the distribution of ξ_t given u_t and x_t

Intuition: an optimal policy over a horizon $\{0, \ldots, H\}$ is optimal for $\{t, \ldots H\}$

Value functions $V_t(x_t)$ allow decomposition of multi-period problem to single-stage optimization problems

Q Function

Define **Q functions**:

$$Q_t(x_t, u_t) = \mathbb{E}_{\xi_t}[c_t(x_t, u_t, \xi_t) + V_{t+1}(f_t(x_t, u_t, \xi_t))|x_t, u_t]$$

Interpretation of $Q_t(x_t, u_t)$: cost of being in x_t given that action u_t has been selected

Value function as a function of Q function:

$$V_t(x_t) = \min_{u_t \in A_t(x_t)} Q_t(x_t, u_t)$$

Curse of Dimensionality

Consider discretization of each component of $x_t \in \mathbb{R}^m$, $u_t \in \mathbb{R}^n$, $\xi_t \in \mathbb{R}^p$ into d points

At stage t, computation of $V_t(x_t)$ for all x_t requires

- for all d^m possible values of x_t
- compute expectation $\Rightarrow$ summation over d^p values of ξ_t
- minimization $\Rightarrow$ comparison of d^n possible values of u_t

Each stage of DP algorithm requires $O(d^{m+n+p})$ operations $\Rightarrow$ overall complexity of $O(H \cdot d^{m+n+p})$

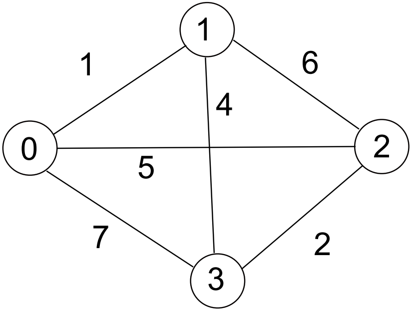
Table of Contents

Recall that central entity of DP algorithm is the *value function*

Main idea of DP efficiency: avoid unnecessary repetition of computation by storing future cost data in value functions

Traveling Salesman Problem

Goal: starting from city 0, find *minimum distance* tour that goes through all cities *exactly once* and returns to 0



c_{ij}: distance from city i to city j (indicated on arcs)

Enumeration

Tour	Distance
01320	12
02310	12
01230	16
03210	16
02130	22
03120	22

Given $H + 1$ cities:

- must examine $H!$ tours
- Computation of cost of each tour: H summations

Complexity of enumeration: $O(H! \cdot H)$

Dynamic Programming for TSP

Idea: interpret each city as one 'stage' in multi-stage decision

State: information necessary for deciding next move

- R_t: set of cities that still need to be visited
- i: current city

Value function $V_t(R_t, i)$: most efficient way of visiting cities in R_t exactly once, starting from i and ending in 0

$$V_t(R_t, i) = \min_{j \in R_t} c_{ij} + V_{t+1}(R_t - \{j\}, j)$$

Note: $V_t(R_t, i)$ is reused

Complexity of Dynamic Programming for TSP

At stage t, computation of V_t for all i, R_t requires:

- for H different values of i
- for $\binom{H}{H-t}$ different values of R_t
- one minimization over $H - t$ values (size of R_t)

Total number of operations:

- For $V_0(\{1, \ldots, H\}, 0)$: H summations
- For $1 \leq t \leq H$:
  $\sum_{t=1}^{H} H \cdot \binom{H}{H-t} \cdot (H - t + 1) = O(H^2 \cdot 2^H)$

Note: Complexity remains exponential, better than factorial

Where did the computational savings come from?

Value function	Evaluation
$V_3(\emptyset, 1)$	1
$V_3(\emptyset, 2)$	5
$V_3(\emptyset, 3)$	7
$V_2(\{2\}, 1) = c_{12} + V_3(\emptyset, 2)$	11
$V_2(\{3\}, 1)$	11
$V_2(\{1\}, 2)$	7
$V_2(\{3\}, 2)$	9
$V_2(\{1\}, 3)$	5
$V_2(\{2\}, 3)$	7
$V_1(\{2, 3\}, 1) = \min\{c_{12} + V_2(\{3\}, 2), c_{13} + V_2(\{2\}, 3)\}$	11
$V_1(\{1, 3\}, 2)$	7
$V_1(\{1, 2\}, 3)$	9
$V_0(\{1, 2, 3\}, 0) = \min_{x \in \{1,2,3\}}(c_{0x} + V_2(\{1, 2, 3\} - \{x\}, x)$	12

Table of Contents

The Knapsack Problem

Consider a back with space limit W and n items. Denote

- v_i: benefit of item i
- $w_i > 0$: volume of item i
- x_i: indicator whether item i is chosen or not

$$
\begin{aligned}
p^\star = \max_x \sum_{i=1}^n v_i x_i \\
\sum_{i=1}^n w_i x_i \leq W \\
x_i \in \{0, 1\} \quad i = 1, \ldots, n,
\end{aligned}
$$

Suppose all data (W, w_i, v_i) is integer

Complexity of Enumeration

Suppose that

- Summing two numbers: one unit of time
- Taking the minimum of two numbers: negligible

For every combination of items (there are 2^n):

- Computation of $\sum_{i=1}^{n} w_i x_i$ requires $n - 1$ summations
- Computation of $\sum_{i=1}^{n} v_i x_i$ requires $n - 1$ summations

Run time of complete enumeration: $2^n \cdot (2n - 2)$

Definition of Value Function

Value function $V(i, w)$:

- Domain: $\{0, \ldots, , n\} \times \{0, \ldots, W\}$
- Interpretation: *best* possible value for a knapsack with capacity w and to which items from the set $\{0, \ldots, i\}$ can be inserted
- Boundary values:
  - $V(0, w) = 0$ (interpretation: no items to include, therefore zero value)
  - $V(i, 0) = 0$ (interpretation: no space in the knapsack, therefore zero value)

Dynamic Programming Principle of Optimality

$$V(i, w) = \max\{V(i-1, w-w_i) + v_i, V(i-1, w)\}$$

- First argument of max operator: include item i in the knapsack
- Second argument of max operator: do not include item i in the knapsack

Dynamic Programming Algorithm

```
for i = 1 : n
  for w = 1 : W
    V(i, w) = max{V(i − 1, w − −w_i) + v_i, V(i − 1, w)};
  end
end
```

- Number of operations: $2 \cdot n \cdot W$
  - Each evaluation of $V(i, w)$ requires two time units
  - Repeat $n \cdot W$ times
- Value of knapsack: $p^\star = V(n, W)$

Items Entering the Knapsack

$K(i)$: 1 if item i is included, 0 otherwise

```
w ← W;
for i = n : 1 do
    K(i) ← 0;
    if v_i + V(i − 1, w − w_i) ≥ V(i − 1, w) then
        K(i) ← 1;
        w ← w − w_i;
    end if
end for
```

The total run time: $3 \cdot n$

Example: Knapsack Problem

Consider $W = 7$ and the following list of items

i	1	2	3	4
v_i	10	40	30	50
w_i	5	4	6	3

Example: Knapsack Problem - Value Function Table

Entry (i, j) corresponds to $V(i, j)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	10	10	10
2	0	0	0	0	40	40	40	40
3	0	0	0	0	40	40	40	40
4	0	0	0	50	50	50	50	90

Example: Knapsack Problem - Items Entering

- For $n = 4$, we have $v_4 + V(3, 7 - w_4) = 50 + V(3, 4) = 50 + 40 \geq V(4, 7) = 90$ $\Rightarrow$ item 4 is included
- For $n = 3$, we have $v_3 + V(2, 3 - w_3) = 30 + V(2, -3) = -\infty < V(3, 4) \Rightarrow$ item 3 is included
- (Check that) item 2 is included
- (Check that) item 1 is not included

The Monty Hall Problem

The following situation emerges in the TV show '*Let's Make a Deal*':

1. A player is asked to pick a curtain
2. The host opens up a curtain with a goat behind it
3. The player can keep the curtain that she chose originally, or switch to the remaining curtain
4. The player keeps the content behind the curtain that was selected in step (iii)

Should the player change curtains in step (iii), or not?

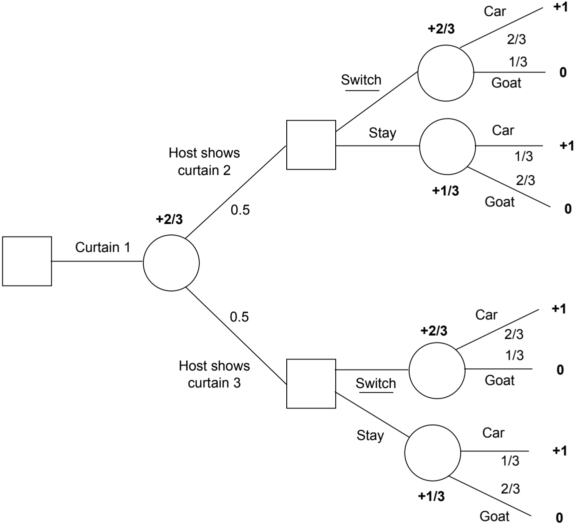
Solution of the Monty Hall Problem

Assumption: when the host opens a curtain in step (ii), the host will choose curtains with equal likelihood if both of the curtains not chosen by the player hide a goat

Thinking about the decision tree:

- We need three stages
- Symmetry $\Rightarrow$ we do not lose generality by assuming that the player picks curtain 1 in the first step
- Symmetry + assumption $\Rightarrow$ equal probability of host opening curtain 2 or curtain 3
- Uncertainty in stage 1 is *not* the location of the sports car, this cannot be observed!
- Compute transition probabilities of second stage using Bayes' theorem

Decision Tree of the Newsboy Problem



Probability of Winnning if We Stay

Bayes' theorem:

$$
\begin{aligned}
&\mathbb{P}[\text{Car in C1}|\text{Host shows C2}] = \\
&\frac{\mathbb{P}[\text{Car in C1, Host shows C2}]}{\mathbb{P}[\text{Host shows C2}]} = \\
&\frac{\mathbb{P}[\text{Host shows C2}|\text{Car in C1}] \cdot \mathbb{P}[\text{Car in C1}]}{\mathbb{P}[\text{Host shows C2}]} = \\
&\frac{1/2 \cdot 1/3}{1/2} = \frac{1}{3}
\end{aligned}
$$

Probability of winning if we stay = original probability of winning

$\Rightarrow$ we have not gained (or lost) anything by staying

Intuitive? Probably ...

Probability of Winnning if We Switch

Bayes' theorem:

$$\mathbb{P}[\text{Car in C3}|\text{Host shows C2}] = \frac{\mathbb{P}[\text{Car in C3, Host shows C2}]}{\mathbb{P}[\text{Host shows C2}]} = \frac{\mathbb{P}[\text{Host shows C2}|\text{Car in C3}] \cdot \mathbb{P}[\text{Car in C3}]}{\mathbb{P}[\text{Host shows C2}]} = \frac{1 \cdot 1/3}{2/3} = \frac{2}{3}$$

Chances of winning double if we switch!

Intuitive? Try this: the host <u>deliberately</u> leaves one door unrevealed

Thinking About the Solution

Two wrong ways to think about the probability of winning if we switch

- Switching is like picking one of three doors $\Rightarrow$ $\mathbb{P}[\text{Car}|\text{Switch}] = 1/3$
- Switching is like picking one of the two leftover doors $\Rightarrow$ $\mathbb{P}[\text{Win}|\text{Switch}] = 1/2$

Pricing Financial Securities

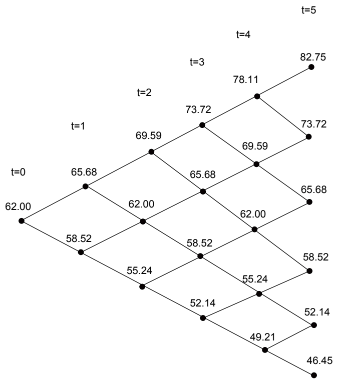
American call option: financial instrument that allows its owner to buy a certain financial asset at a *strike price* at <u>or before</u> a certain *expiration date*

Call option at time t is worth $\max(S_t - k, 0)$, where

- S_t: price of financial asset at time t
- k: strike price of the option

Use DP in order to determine how much an option is worth at time $t = 0$

Lattice Model of Stock Price S_t



Consider the following transition probabilities:

- upward: $q = 0.5577$
- downward: $1 - q$

Backward Solution

Denote $V_t(i)$ as the value of the option at stage t, and state i, where i corresponds to one of the nodes in the lattice

Consider strike price $k = 60$

Period 5 payoff:

$$V_5(1) = 82.75 - 60 = 22.75$$
$$V_5(2) = 73.72 - 60 = 13.72$$
$$V_5(3) = 65.68 - 60 = 5.68$$
$$V_5(4) = 0$$
$$V_5(5) = 0$$
$$V_5(6) = 0$$

Backward Solution (II)

Period 4 payoff:

$$V_4(1) = \max(78.11 - 60, \mathbb{E}[V_5(j)|i = 1]) = 18.7560$$

$$V_4(2) = \max(69.59 - 60, \mathbb{E}[V_5(j)|i = 2]) = 10.1639$$

$$V_4(3) = \max(62 - 60, \mathbb{E}[V_5(j)|i = 3]) = 3.1677$$

$$V_4(4) = 0$$

$$V_4(5) = 0$$

Period 3 payoff:

$$V_3(1) = \max(73.72 - 60, \mathbb{E}[V_4(j)|i = 1]) = 14.9557$$

$$V_3(2) = \max(65.68 - 60, \mathbb{E}[V_4(j)|i = 2]) = 7.0695$$

$$V_3(3) = \max(0, \mathbb{E}[V_4(j)|i = 3]) = 1.7666$$

$$V_3(4) = 0$$

Backward Solution (III)

Period 2 payoff:

$$V_2(1) = \max(69.59 - 60, \mathbb{E}[V_3(j)|i = 1]) = 11.4676$$
$$V_2(2) = \max(62 - 60, \mathbb{E}[V_3(j)|i = 2]) = 4.7240$$
$$V_2(3) = 0.9852$$

Period 1 payoff:

$$V_1(1) = \max(65.68 - 60, \mathbb{E}[V_2(j)|i = 1]) = 8.4849$$
$$V_1(2) = \max(0, \mathbb{E}[V_2(j)|i = 2]) = 3.0703$$

Period 0 payoff:

$$V_0(1) = \max(62 - 60, \mathbb{E}[V_1(j)|i = 1]) = 6.09$$

Optimal Exercise Policy

Note that in the previous example the optimal policy was to hold on to the American call option

We will now prove that this was not a coincidence

General Model of Asset Price Evolution

Consider the following asset price model: given asset price S_t at stage t

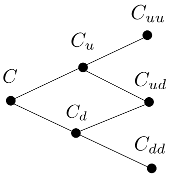
- upward transition in $t+1$ with probability q results in stock price $u \cdot S_t$ with $u > 1$
- downward transition in $t+1$ with probability $1-q$ results in stock price $d \cdot S_t$ with $d < 1$

Choose q so that expected value of asset does not change:

$$q = \frac{1-d}{u-d}$$

Optimal Exercise Policy of American Call Option

We will show that it is optimal to *hold on* to American call option *until expiration*



Using induction, it suffices to show that $C_u \geq u \cdot S - k$ and $C_d \geq d \cdot S - k$, where

- C: value of call option at stage t
- C_u / C_d: value of call option at stage $t + 1$
- $C_{uu} / C_{ud} / C_{dd}$: value of call option at stage $t + 2$

Optimal Exercise Policy of American Call Option

Value of waiting at top node of stage $t+1$:

$$\begin{aligned} C_u &= q \cdot C_{uu} + (1-q) \cdot C_{ud} \\ &\geq q \cdot \max(u^2 \cdot S - k, 0) + (1-q) \cdot \max(u \cdot d \cdot S - k, 0) \\ &\geq \max(q(u^2 \cdot S - k) + (1-q) \cdot (u \cdot d \cdot S - k), 0) \\ &= \max(u \cdot S - k, 0) \\ &\geq u \cdot S - k \end{aligned}$$

- First inequality: expected payoff in stage $t+2$ at least as much as exercising the option in stage $t+2$
- Second inequality: convexity of $\max(x, 0)$

Same reasoning $\Rightarrow C_d \geq d \cdot S - k$