Dynamic programming

An alternative approach to dynamic optimisation

Rolf Groeneveld

Unknowns in natural resource management

- State variables
  - How much fish is in the sea?
  - How much oil is in the ground?
- Flow variables
  - How fast will fish stocks grow?
  - How much do fishers *really* catch?
  - How much rain will fall?

Dealing with stochastic problems

- Backward induction
  - Tedious but intuitive
- Dynamic programming
  - Analytical: suitable for stochastic problems but hard
  - Numerical: widely applied to stochastic problems

Learning outcomes

- After this lecture you are expected to be able to
  - Draw a simple decision tree of a dynamic optimization problem
  - Formulate the Bellman Equation of a given optimization problem in discrete time and in continuous time
  - Write a simple dynamic programming model in R
  - Explain the relation between dynamic programming and optimal control

Program

- How does risk change dynamic optimization?
- Backward induction
- Dynamic Programming
  - Analytic
  - Numerical

How does risk change dynamic optimization?

- Optimal Control
  - Optimal extraction path: $Y^*(t)$
- Risk
  - You get 'blown off the path' all the time
  - The plan does not tell you what to do if you stray
  - So very soon your planned route is not optimal anymore

How does risk change dynamic optimization?

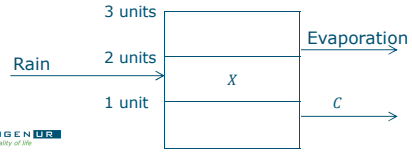
- Adaptive management
  - Make plans
  - But be prepared to change them
- Dynamic Programming
  - Optimal extraction path: $Y^*(X,t)$ or even $Y^*(X)$

Program

- How does risk change dynamic optimization?
- Backward induction
- Dynamic Programming
  - Analytic
  - Numerical

A simple example

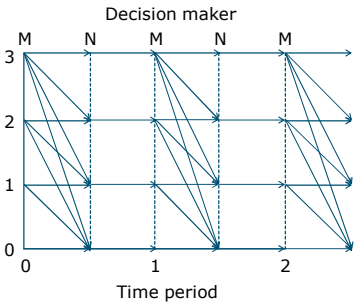
- Consider a tub of water used for irrigating crops
- State variable: water stock (X)
- Control variable: water consumption (C)
- Immediate utility: $\sqrt{C}$
- State equation: $\Delta X = -C + z$
  - Where z denotes net replenishment
- Discount rate $r = 0.1$



Decision tree of water use

- Imagine this is a game against 'nature'
  - Player 1: Man (three moves)
  - Player 2: Nature (two moves)
- Versions
  - Nature *always* plays $z = 0$ (cake-eating problem)
  - Nature *always* plays $z = 1$
  - Nature plays $z = 1$ or $z = -1$ (stochastic problem)

Decision tree for cake-eating problem

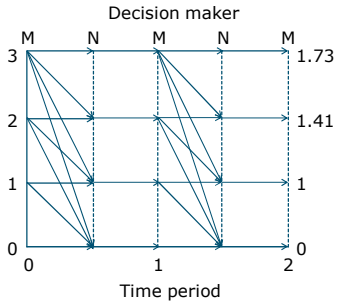


Backward induction

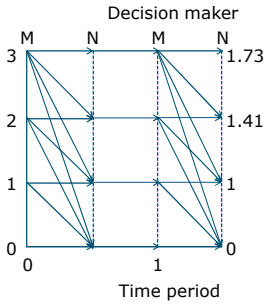
- Start at the last time period ($t = T$)
- How much would we consume if it was our last day on the farm?

Stock (X)	Consumption (C)	Utility ($\sqrt{C}$)
0	0	0
1	1	1
2	2	1.41
3	3	1.73

Decision tree for cake-eating problem



Decision tree for cake-eating problem



Backward induction

- Go to the year before last ($t = T - 1$)
- We now face a trade-off between now and next period:

$$\max_{C}\left\{\sqrt{C} + \frac{1}{1+r}\sqrt{X-C}\right\}$$

- For example, if $X = 2$:

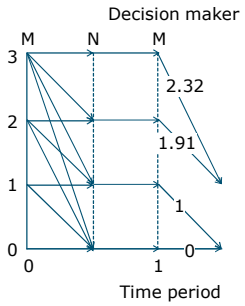
C	$\sqrt{C}$	$\sqrt{X-C}$	$\sqrt{C} + \frac{1}{1+r}\sqrt{X-C}$
0	0	1.41	1.29
1	1	1	1.91
2	1.41	0	1.41
3	-	-	-

Backward induction

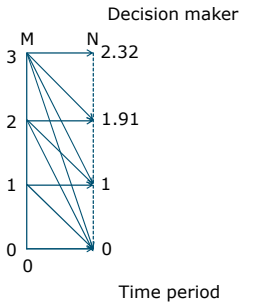
- So for each combination of stock size and consumption we can calculate the total utility and pick the best level:

X	$C = 0$	$C = 1$	$C = 2$	$C = 3$
0	0			
1	0.91	1		
2	1.29	1.91	1.41	
3	1.57	2.29	2.32	1.73

Decision tree for cake-eating problem



Decision tree for cake-eating problem

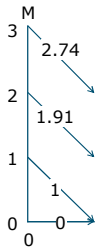


Backward induction

- For each combination of stock size and consumption we again calculate the total utility and pick the best level:

X	$C=0$	$C=1$	$C=2$	$C=3$
0	0			
1	0.91	1		
2	1.74	1.91	1.41	
3	2.11	2.74	2.32	1.73

Decision tree for cake-eating problem



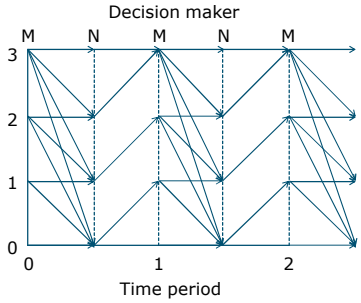
Backward induction

- Optimal consumption path is a function of
  - Stock size
  - Number of remaining time periods
- This is also called a 'policy function'

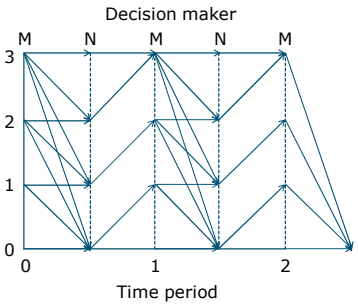
  $C = C(X,t)$
- As the time horizon goes to infinity the policy function depends on stock size only

  $C = C(X)$

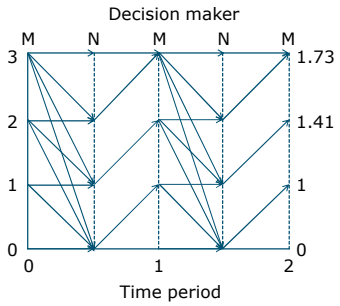
Decision tree if $z = 1$



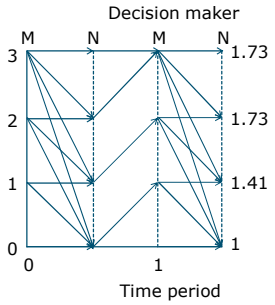
Decision tree if $z = 1$



Decision tree if $z = 1$



Decision tree if $z = 1$



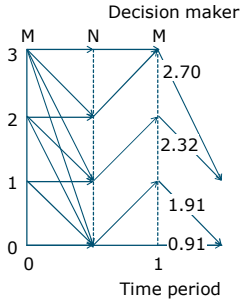
Backward induction

- For each combination of stock size and consumption calculate the total utility and pick the best level:

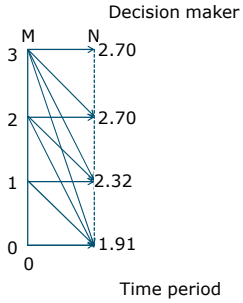
X	$C=0$	$C=1$	$C=2$	$C=3$
0	0.91			
1	1.29	1.91		
2	1.57	2.29	2.32	
3	1.57	2.57	2.70	2.64

- Note that even if the stock is empty we get some discounted future utility

Decision tree if $z = 1$



Decision tree if $z = 1$

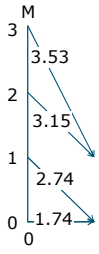


Backward induction

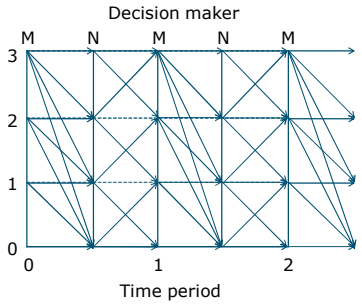
- For each combination of stock size and consumption calculate the total utility and pick the best level:

X	$C=0$	$C=1$	$C=2$	$C=3$
0	1.74			
1	2.11	2.74		
2	2.45	3.11	3.15	
3	2.45	3.45	3.53	3.47

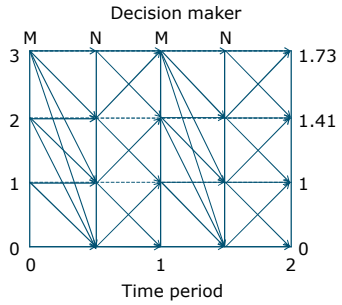
Decision tree if $z = 1$



Decision tree for stochastic problem



Decision tree for stochastic problem

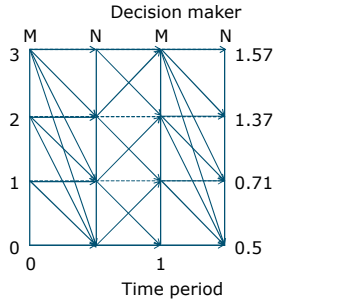


Nature's moves

- We don't know z, but we do have a probability distribution
  - 0.5 probability that $z = 1$
  - 0.5 probability that $z = -1$
- We can therefore calculate the expected value of next period's value

X	Value if $z = -1$	Value if $z = 1$	Expected value
0	0	1	0.5
1	0	1.41	0.71
2	1	1.73	1.37
3	1.41	1.73	1.57

Decision tree for stochastic problem



Backward induction

- We now face a trade-off between now and next period
- But now we have for the next period an expected value:

$$\max_C \left\{ \sqrt{C} + \frac{1}{1+r} E\left(\sqrt{X - C}\right) \right\}$$

- For example, if $X = 2$:

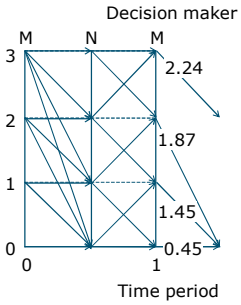
C	$\sqrt{C}$	$E\left(\sqrt{X-C}\right)$	$\sqrt{C} + \frac{1}{1+r} E\left(\sqrt{X-C}\right)$
0	0	1.37	1.24
1	1	0.71	1.64
2	1.41	0.5	1.87
3	-	-	-

Backward induction

- For each combination of stock size and consumption we can calculate the total utility and pick the best level:

X	$C = 0$	$C = 1$	$C = 2$	$C = 3$
0	0.45			
1	0.64	1.45		
2	1.24	1.64	1.87	
3	1.43	2.24	2.06	2.19

Decision tree for stochastic problem

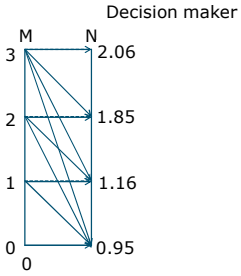


Nature's moves

- Calculate the expected value of next period's value

X	Value if $z=-1$	Value if $z=1$	Expected value
0	0.45	1.45	0.95
1	0.45	1.87	1.16
2	1.45	2.24	1.85
3	1.87	2.24	2.06

Decision tree for stochastic problem



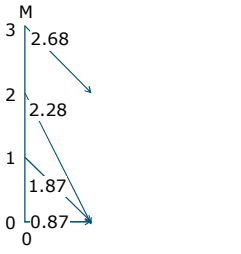
Backward induction

- For each combination of stock size and consumption we can calculate the total utility and pick the best level:

X	$C=0$	$C=1$	$C=2$	$C=3$
0	0.87			
1	1.06	1.87		
2	1.68	2.06	2.28	
3	1.87	2.68	2.47	2.60

- Note that the utility has changed but not the optimal policy

Decision tree for stochastic problem



Stochastic problem

- Note the differences with the other two cases
- $z=0$
  - Expected value of z is equal
  - But if $X=2$ you use one unit more because future payoffs are uncertain
- $z=1$
  - Expected value of z is lower
  - If $X=3$ you save more for the future

Program

- How does risk change dynamic optimization?
- Backward induction
- Dynamic Programming
  - Analytic
  - Numerical

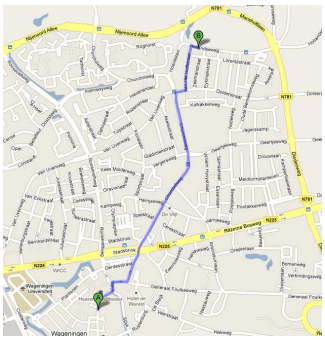
Dynamic programming

- We just applied Bellman's Principle of Optimality:

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

Bellman's principle of optimality

- What this means is that your Tom-Tom never changes your route as you drive



Discrete-time Bellman Equation

$$V_t(X_t) = \max_{C_t}\left\{U(X_t, C_t) + \frac{1}{1+r}V_{t+1}(X_{t+1}(X_t, C_t))\right\}$$

- where
  - $V_t(X_t)$ is the *value function*
  - $U(\blacksquare)$ is the *immediate utility function*

Dynamic programming example: fishery

- Objective: $\max_{Y_t} \sum_{t=0}^{\infty} \frac{pY_t - C(X_t, Y_t)}{(1+r)^t}$
- State equation: $X_{t+1} = X_t + G(X_t) - Y_t$
- Bellman equation:

$$V_t(X_t) = \max_{Y_t}\left\{pY_t - C(X_t, Y_t) + \frac{1}{1+r}V_{t+1}(X_t + G(X_t) - Y_t)\right\}$$

Dynamic programming example: fishery

- Assume a steady state so that
  - $X_t = X_{t+1} = X$
  - $Y_t = Y_{t+1} = Y$
  - $V_t = V_{t+1} = V$
- The Bellman equation then becomes:

$$V(X) = \max_{Y}\left\{pY - C(X, Y) + \frac{1}{1+r}V(X + G(X) - Y)\right\}$$

Dynamic programming example: fishery

$$V(X) = \max_Y \left\{ pY - C(X,Y) + \frac{1}{1+r} V(X + G(X) - Y) \right\}$$

Take first derivative to find condition for optimal harvest:

$$p - C_Y = \frac{1}{1+r} V_X$$

- LHS: marginal benefits of reducing stock by 1 unit
- RHS: marginal benefits of increasing stock by 1 unit

Dynamic programming example: fishery

$$p - C_Y = \frac{1}{1+r} V_X$$

- How do we find an expression for V_X?
- Take first derivative of the entire Bellman equation:

$$V_X = -C_X + \frac{1}{1+r} V_X [1 + G_X]$$

- Is this allowed, despite the maximization of V?
- Yes, it's called the Envelope Theorem

Dynamic programming example: fishery

- Isolate V_X:

$$V_X = \frac{1+r}{G_X - r} C_X$$

- Substitute V_X in the optimality condition:

$$p - C_Y = \frac{1}{G_X - r} C_X \Rightarrow$$

$$[p - C_Y] G_X - C_X = r[p - C_Y]$$

- Compare Lecture 2:

$$(p - C_Y) G_X - C_X = \rho (p - C_Y)$$

Stochastic Bellman Equation

$$V(X_t) = \max_{C_t} \left\{ U(X_t, C_t) + \frac{1}{1+r} E(V(X_{t+1}) | X_t, C_t) \right\}$$

- where
  - $E(\blacksquare)$ means 'expected value of'
- We have some information on the uncertain process
  - A stochastic variable $\tilde{z}$
  - How future stocks X' depend on $\tilde{z}$
  - A probability distribution of $\tilde{z}$

Continuous time Bellman equation

- If we assume that
  - A time period has length Δt
  - Utility U in period t is equal to $u\Delta t$
  - Control C in period t is equal to $c\Delta t$
  - Discrete-time discount rate r is equal to $\rho \Delta t$
- Then we can write the Bellman equation like this:

$$V(X,t) = \max_c \left\{ u(X,c,t)\Delta t + \frac{1}{1+\rho\Delta t} V(X', t+\Delta t) \right\}$$

- Where X' is the value of X in next period

Continuous time Bellman equation

$$V(X,t) = \max_c \left\{ u(X,c,t)\Delta t + \frac{1}{1+\rho\Delta t} V(X', t+\Delta t) \right\}$$

- It is possible to rewrite this to

$$\rho V(X,t) = \max_c \left\{ u(X,c,t)[1+\rho\Delta t] + \frac{V(X', t+\Delta t) - V(X',t)}{\Delta t} \right\}$$

- If you let Δt go to zero this becomes

$$\rho V(X,t) = \max_c \left\{ u(X,c,t) + \frac{dV}{dt} \right\}$$

Program

- How does risk change dynamic optimization?
- Backward induction
- Dynamic Programming
  - Analytic
  - Numerical

Numerical dynamic programming

- The real strength of dynamic programming lies in its numerical application
- Value Function Iteration (similar to backward induction)
  1. Discretize state variables
  2. Identify optimal control for each value
  3. Update value function for the optimal control
  4. Repeat steps 2-3 until convergence

Discretizing state variable

- In our water example we had discretized water stock to four possible levels:

$$\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

Identify optimal control

- In the water example the control was defined by next year's stock size
  - 3 units now: consume 0, 1, 2, or 3 units
  - 2 units now: consume 0, 1, or 2 units
  - 1 unit now: consume 0 or 1 unit
  - 0 units now: consume 0 units
- In a numerical model
  - Try all control values and pick the best
  - Use some built-in optimization procedure

Update value function

- Start with all zeros: $\mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$
- So the first iteration features only the direct utility
- The second iteration features
  - Direct utility plus
  - Next period's (expected) optimal value depending on this year's control
- And so on

Convergence

- How many iterations do you need?
- Finite horizon problems
  - Start at the transversality conditions
  - Iterate (i.e. count back) for each time period
- Infinite horizon problems
  - Initial conditions are further discounted with each iteration
  - Policy function and value function should converge
  - How much convergence is needed is up to you

Computational Dynamic Programming

- Most suitable for dealing with uncertainty
- Drawbacks
  - It takes a lot of programming
  - Inaccuracies due to discretization
  - Curse of dimensionality

WAGENINGEN UR
For quality of life

Tomorrow

8:30 – 10:15 in PC621
Computer practical
Dynamic Programming

10.30 – 12.15
Nonlinearities and complexity in ecosystems

WAGENINGEN UR
For quality of life