MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Civil and Environmental Engineering

1.731 Water Resource Systems

Lecture 19,20 Real-Time Optimization, Dynamic Programming
Nov. 14, 16, 2006

Real-time optimization

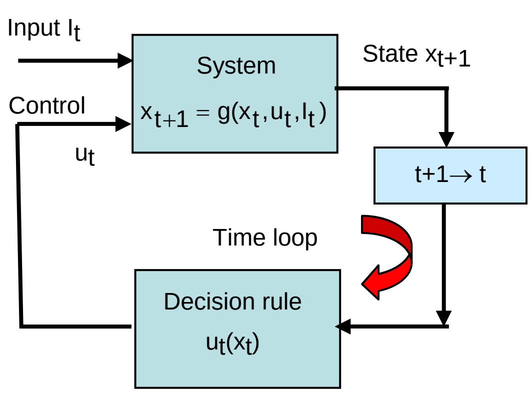
Real-time optimization problems rely on **decision rules** that specify how decisions should maximize future benefit, given the current **state** of a system. State dependence provides a convenient way to deal with **uncertainty**. Some examples:

- Reservoir releases – Decision rule specifies how current release should depend on current storage. Primary uncertainty is future reservoir inflow.

- Water treatment – Decision rule specifies how current operating conditions (e.g. temperature or chemical inputs) should depend on current concentration in treatment tank. Primary uncertainty is future influent concentration.

- Irrigation management - Decision rule specifies how current applied irrigation water should depend on current soil moisture and temperature. Primary uncertainties are future meteorological variables.

Real-time optimization can be viewed as a feedback control process:



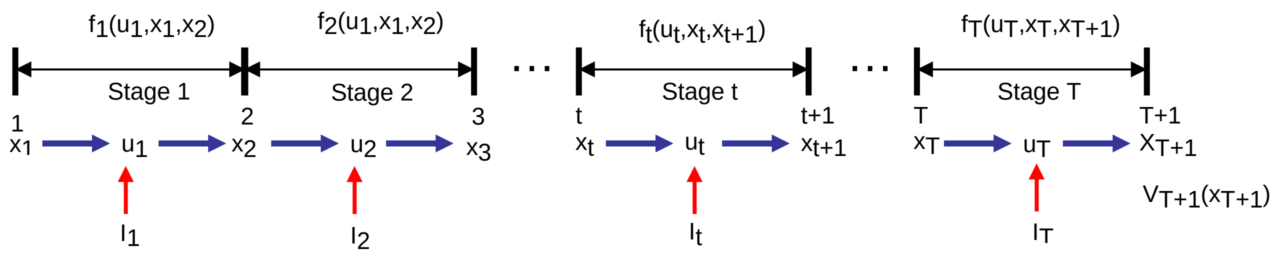
At each time step:
- Observe state
- Derive control from decision rule
- Apply control.

State variables:	x_t	(decision variables, depend on controls and inputs)
Control variables:	u_t	(decision variables, selected to maximize benefit)
Input variables:	I_t	(inputs, assigned specified values)
Decsion rule:	$u_t(x_t)$	(function that gives u_t for any x_t)

State equation: $x_{t+1} = g(x_t, u_t, I_t)$

Dynamic Programming

Dynamic programming provides a general framework for deriving decision rules. Discrete dynamic programming divides problem into stages (e.g. time periods, spatial intervals, etc.):



Benefit accrued over Stage t is $f_t(u_t, x_t, x_{t+1})$:

Optimization problem:

Select $u_t,...,u_T$ that **maximizes benefit-to-go** (benefit accrued from current time t through terminal time T+1) at each time t:

$$\underset{u_t,...u_T}{Max}\ F_t(x_t,...,x_{T+1},u_t,...,u_T) \qquad t = 1,\ldots,T$$

where benefit-to-go at t is terminal benefit (**salvage value**) $V_{T+1}(x_{T+1})$ plus sum of benefits for stages t through T:

$$: \ F_t(x_t,...,x_{T+1},u_t,...,u_T) = \sum_{i=t}^{T} f_i(u_i, x_i, x_{i+1}) \quad + \quad V_{T+1}(x_{T+1})$$

Benefit-to-go **Benefit from remaining stages** **Terminal benefit**

subject to:

$$x_{i+1} = g_t(x_i, u_i, I_i) \quad ; \quad i = t,...,T \quad \text{(state equation)}$$

and other constraints on the decision variables:

$\{x_t,..., x_{T+1}, u_t,..., u_T\} \in \Gamma_t$; $i = t,...,T$ (decision variables lie within some set Γ_t at t).

Objective may be rewritten if we repeatedly apply state equation to write all x_i $(i > t)$ as functions of $x_t, u_t,..., u_T, I_t,..., I_T$:

$$F_t(x_t,..., x_{T+1}, u_t,..., u_T) = F_t(x_t, u_t,..., u_T, I_t,..., I_T)$$

Decision rule $u_t(x_t)$ at each t is obtained by finding sequence of controls $u_t,..., u_T$ that maximizes $F_t(x_t, u_t,..., u_T, I_t,..., I_T)$ for a given state x_t and a given set of specified inputs $I_t,..., I_T$.

Backward Recursion for Benefit-to-go

Dynamic programming uses a **recursion** to construct decision rule $u_t(x_t)$:

Define **return function** $V_t(x_t)$ to be maximum benefit-to-go at *t*:

$$V_t(x_t) = \underset{u_t,\ldots,u_T}{Max} \left[F_t(x_t, u_t, \ldots, u_T, I_t, \ldots, I_T)\right]$$

Separate benefit term for Stage *t*:

$$V_t(x_t) = \underset{u_t}{Max}\left[f_t(u_t, x_t, x_{t+1}) + \underset{u_{t+1},\ldots u_T}{Max} F_{t+1}(x_{t+1}, u_{t+1}, \ldots, u_T, I_{t+1}, \ldots, I_T) \right]$$

Replace second term in brackets with definition for $V_{t+1}(x_{t+1})$:

$$V_t(x_t) = \underset{u_t}{Max}\left[f_t(u_t, x_t, x_{t+1}) + V_{t+1}(x_{t+1})\right]$$

Substitute state equation for x_{t+1}:

$$V_t(x_t) = \underset{u_t}{Max}\left\{f_t[u_t, x_t, g_t(x_t, u_t, I_t)] + V_{t+1}[g_t(x_t, u_t, I_t)]\right\}$$

This equation is a **backward recursion** for $V_t(x_t)$, initialized with terminal benefit $V_{T+1}[g(x_t, u_t, I_t)]$. Expression in braces depends only on u_t [which is varied to find the maximum], x_t [the argument of $V_t(x_t)$], and I_t [the specified input].

At each stage find the u_t that maximizes $V_t(x_t)$ for all possible x_t.
Store the results (e.g. as a function or table) to obtain the desired decision rule $u_t(x_t)$.

Computational Effort

The problem variables are frequently **discretized** into a finite number of values x_t^j, u_t^j, I_t^j , j = 1, 2, …,L where L =number of discrete levels.

The solution to the discretized optimization problem can be found by **exhaustive enumeration** (by comparing benefit-to-go for all possible $u_t(x_t)$ combinations).

Dynamic programming is much more efficient than enumeration since it divides the original T stage optimization problem into T smaller problems, one for each stage.

To compare computational effort of enumeration and dynamic programming assume:

- State dimension = M, Stages = T, Levels = L
- Equal number of levels for u_t and x_t at every stage
- All possible state transitions are permissible (i.e. L^2 transitions at each stage)

Then total number of V_t evaluations required is:

Exhaustive enumeration: $L^{M(T+1)}$

Dynamic Programming: TL^{2M}

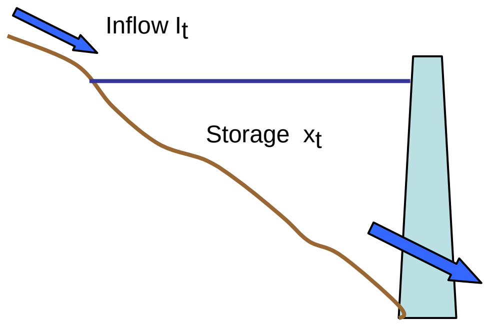
For M = 1, L = 10, T = 10 the number of V_t evaluations required is:

Exhaustive enumeration: 10^{11}

Dynamic Programming: 10^3

Example 1: Reservoir Operations

Maximize benefits from water released from reservoir with variable inflows.
Stages correspond to 3 time periods (months, seasons, etc. T = 3).



State equation:

$$x_{t+1} = x_t - u_t + I_t \qquad t = 1,\ldots,3$$

Total benefit from released water and final storage x_4:

$$F(x_1, u_1, \ldots, u_3) = f_1(u_1) + f_2(u_2) + f_3(u_3) + V_4(x_4)$$

Discretize all variables into consistent levels:

u_t = {0,1,2} x_t = {0,1,2) I_t = {0,1) t = 1, 2, 3

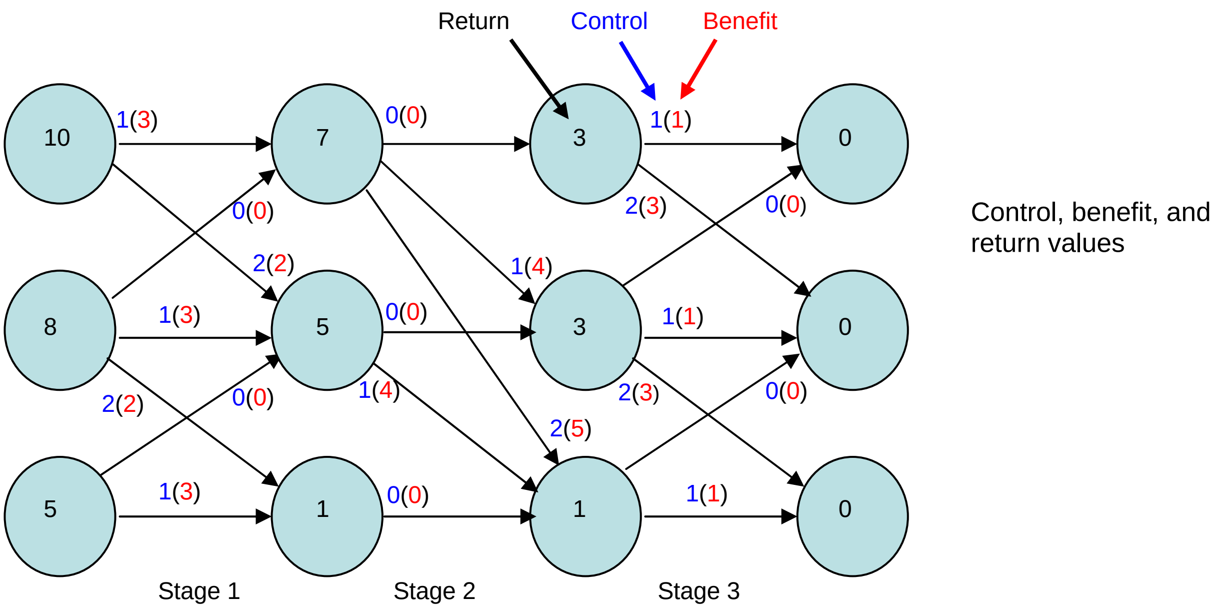
Inflows:	I_1	I_2	I_3
	1	0	1

Terminal (outflow) benefits: $V_4(x_4)$ = 0 for all x_4 values

Benefits for each release:

u_t	$f_1(u_1)$	$f_2(u_2)$	$f_3(u_3)$
0	0	0	0
1	3	4	1
2	2	5	3

Possible state transitions are derived from state equation, inputs, and permissible variable values:
Benefit is shown in parentheses after each feasible control value.



Solve series of 3 optimization problems defined by recursion equation for t = 3, 2, 1.
Start at last stage and move backward:

Stage 3: Maximize $V_2(x_2)$ for each level of x_3:

$$V_3(x_3) = \underset{u3}{Max}\left[f_3(u_3) + V_4(x_4)\right] = \underset{u3}{Max}\left[f_3(u_3) + V_4(x_3 - u_3 + I_3)\right]$$

Identify optimum $u_3(x_3)$ values for each x_3, $V_4(x_4)$ specified as an input:

x_3	$u_3(x_3)$	$f_3(u_3)$	+	$V_4(x_4)$		
0	0	0	+	0	= 0	
	1	1	+	0	= 1 = $V_3(x_3)$	← Optimum
1	0	0	+	0	= 0	
	1	1	+	0	= 1	
	2	3	+	0	= 3 = $V_3(x_3)$	← Optimum
2	1	1	+	0	= 1	
	2	3	+	0	= 3 = $V_3(x_3)$	← Optimum

Stage 2: Maximize $V_2(x_2)$ for each level of x_2:

$$V_2(x_2) = \underset{u_2}{Max}\left[f_2(u_2) + V_3(x_3)\right] = \underset{u_2}{Max}\left[f_2(u_2) + V_3(x_2 - u_2 + I_2)\right]$$

Identify optimum $u_2(x_2)$ value for each x_2, obtain $V_3(x_3)$ from Stage 3:

x_2 $u_2(x_2)$ $f_2(u_2)$ + $V_3(x_3)$

0	0	0	+	1	= 1 = $V_2(x_2)$	⟵ Optimum
1	0	0	+	3	= 3	
	1	4	+	1	= 5 = $V_2(x_2)$	⟵ Optimum
2	0	0	+	3	= 4	
	1	4	+	3	= 7 = $V_2(x_2)$	⟵ Optimum
	2	5	+	1	= 6	

Stage 1: Maximize $V_1(x_1)$ for each level of x_1:

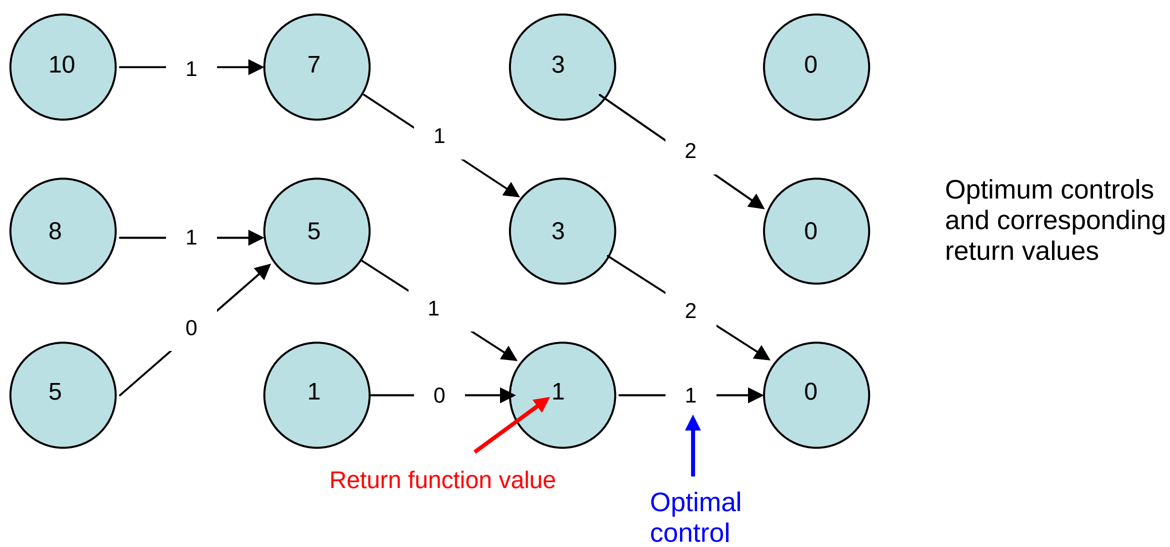
$$V_1(x_1) = \underset{u_1}{Max}\left[f_1(u_1) + V_2(x_2)\right] = \underset{u_1}{Max}\left[f_1(u_1) + V_2(x_1 - u_1 + I_1)\right]$$

Identify optimum $u_1(x_1)$ values for each x_1, obtain $V_2(x_2)$ from Stage 2:

x_1	$u_1(x_1)$	$f_2(u_2)$	+	$V_2(x_2)$		
0	0	0	+	5	= 5 = $V_1(x_1)$	⟵ Optimum
	1	3	+	1	= 4	
1	0	0	+	7	= 7 = $V_1(x_1)$	⟵ Optimum
	1	3	+	5	= 8	
	2	2	+	1	= 3	
2	1	3	+	7	= 10 = $V_1(x_1)$	⟵ Optimum
	2	2	+	5	= 7	

The optimum $u_t(x_t)$ decision rules for t = 1, 2, 3 define a complete optimum decision strategy:

x_1	u_1	x_2	u_2	x_3	u_3
0	0	0	0	0	1
1	1	1	1	1	2
2	1	2	1	2	2



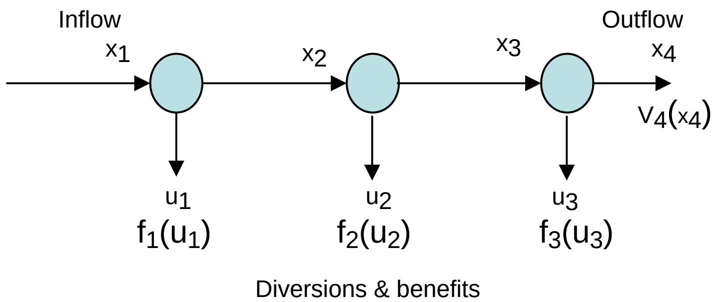
Note that there is a path leaving every state value. The optimum paths give a strategy for maximizing benefit-to-go from t onward, for any value of state x_t.

Optimal benefit for each possible initial storage is $V_1(x_1)$.

Example 2: Aqueduct diversions
Maximize benefits from water diverted from 3 locations along an aqueduct.
Here the stages correspond to aqueduct sections rather than time ($T = 3$).

State equation:

$$x_{t+1} = x_t - u_t \qquad t = 1,\ldots,3$$



Total benefit from diverted water and outflow x_4:

$$F(x_1, u_1, ..., u_3) = f_1(u_1) + f_2(u_2) + f_3(u_3) + V_4(x_4)$$

Discretize all variables into 3 levels:

$$u_t = \{0,1,2\} \qquad x_t = \{0,1,2) \quad t = 1, 2, 3$$

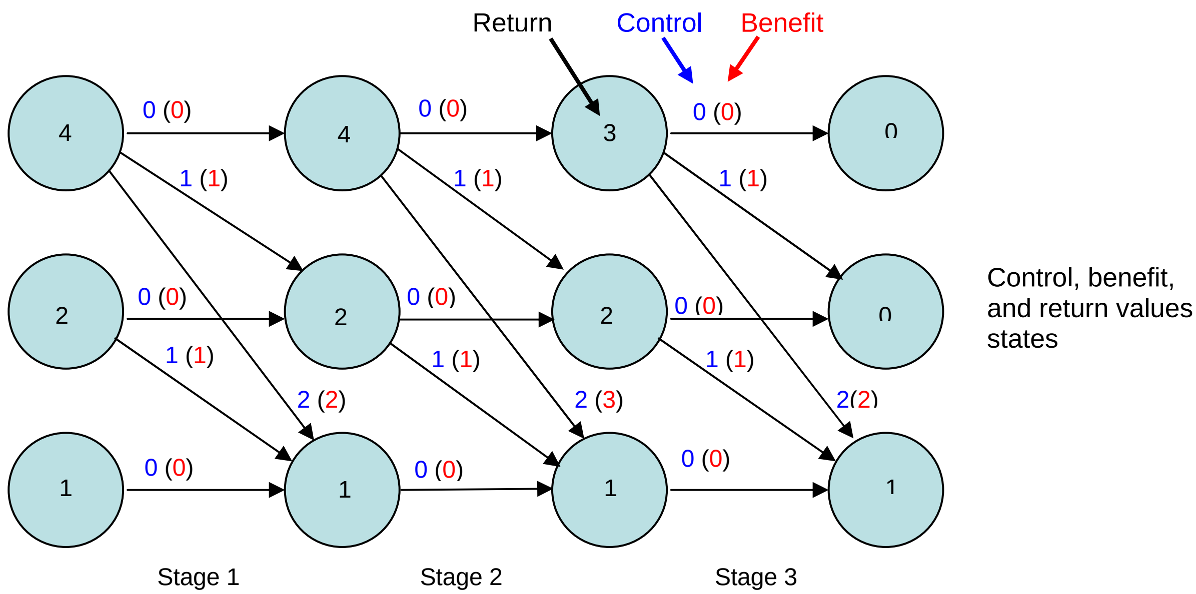
Benefits for each diversion:

u_t	$f_1(u_1)$	$f_2(u_2)$	$f_3(u_3)$
0	0	0	0
1	1	1	1
2	2	3	2

Terminal (outflow) benefits:

x_4	$V_4(x_4)$
0	1
1	0
2	0

Possible state transitions are derived from state equation, inputs, and permissible variable values: Benefit is shown in parentheses after each feasible control value.



Solve series of 3 optimization problems defined by recursion equation for t = 3, 2, 1. Start at last stage and move backward.

Stage 3: Maximize $V_2(x_2)$ for each level of x_3:

$$V_3(x_3) = \underset{u3}{Max}\left[f_3(u_3) + V_4(x_4)\right] = \underset{u3}{Max}\left[f_3(u_3) + V_4(x_3 - u_3)\right]$$

Use same procedure as in Example 1. Identify optimum $u_3(x_3)$ value for each x_3, $V_4(x_4)$ specified as an input. Resulting optimum controls and returns are:

x_3	$u_3(x_3)$	$V_3(x_3)$
0	0	1
1	1	2
2	2	3

Stage 2: Maximize $V_2(x_2)$ for each level of x_2:

$$V_2(x_2) = \underset{u_2}{Max}\left[f_2(u_2) + V_3(x_3)\right] = \underset{u_2}{Max}\left[f_2(u_2) + V_3(x_2 - u_2)\right]$$

Use same procedure as in Example 1. Identify optimum $u_2(x_2)$ value for each x_2, obtain $V_3(x_3)$ from Stage 3. Resulting optimum controls and returns are:

x_2	$u_2(x_2)$	$V_2(x_2)$
0	0	1
1	0 or 1	2
2	2	4

Stage 1: Maximize $V_1(x_1)$ for each level of x_1:

$$V_1(x_1) = \underset{u_1}{Max}\left[f_1(u_1) + V_2(x_2)\right] = \underset{u_1}{Max}\left[f_1(u_1) + V_2(x_1 - u_1)\right]$$

Use same procedure as in Example 1. Identify optimum $u_1(x_1)$ value for each x_1, obtain $V_2(x_2)$ from Stage 2. Resulting optimum controls and returns are:

x_1	$u_1(x_1)$	$V_1(x_1)$
0	0	1
1	0 or 1	2
2	0	4

The optimum $u_t(x_t)$ decision rules for t = 1, 2, 3 define a complete optimum decision strategy:

x_1	u_1	x_2	u_2	x_3	u_3
0	0	0	0	0	0
1	0 or 1	1	0 or 1	1	1
2	0	2	2	2	2

Optimal benefit for each possible inflow is $V_1(x_1)$.