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.

\[ t = 0, \ldots, T - 1 \]

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)
Decision rule: \( u_t(x_t) \) (function that gives \( u_t \) for any \( x_t \))
State equation: \( x_{t+1} = g(x_t, u_t, I_t) \) initial condition: \( x_0 \)
Dynamic Programming
Dynamic programming provides a general framework for deriving decision rules. Most dynamic programming problems are divided into stages (e.g. time periods, spatial intervals, etc.):

Stage 0

\[ u_0 \ x_0 \ x_1 \rightarrow f_0(u_0, x_0, x_1) \]

\[ I_0 \]

Stage 1

\[ u_1 \ x_1 \ x_2 \rightarrow f_1(u_1, x_1, x_2) \]

\[ I_1 \]

Stage t

\[ u_{t-1} \ x_{t-1} \ x_t \rightarrow f_{t-1}(u_{t-1}, x_{t-1}, x_t) \]

\[ I_{t-1} \]

Stage T

\[ u_{T-1} \ x_{T-1} \ x_T \rightarrow f_{T-1}(u_{T-1}, x_{T-1}, x_T) \]

\[ I_{T-1} \]

\[ V_T(x_T) \]

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

Optimization problem:
Select \( u_t, ..., u_{T-1} \) that maximizes benefit-to-go (benefit accrued from current time \( t \) through terminal time \( T \)) at each time \( t \):

\[
\max_{u_t, ..., u_{T-1}} F_t(x_t, ..., x_{T-1}, u_t, ..., u_{T-1}) \quad t = 0, ..., T-1
\]

where benefit-to-go at \( t \) is terminal benefit (salvage value) \( V_T(x_T) \) plus sum of benefits for stages \( t \) through \( T-1 \):

\[
F_t(x_t, ..., x_{T-1}, u_t, ..., u_{T-1}) = \sum_{i=t}^{T-1} f_i(u_i, x_i, x_{i+1}) + V_T(x_T)
\]

subject to:
\[
x_{i+1} = g_i(x_i, u_i, I_i) \quad ; \quad i = t, ..., T-1 \quad \text{(state equation)}
\]

and other constraints on the decision variables:
\[
\{x_t, u_t\} \in \Gamma_t, \quad \{x_T\} \in \Gamma_T \quad \text{(decision variables lie within some set } \Gamma_t \text{ at } t = 0, ..., T)\]

Objective may be rewritten if we repeatedly apply state equation to write all \( x_i \) \((i > t)\) as functions of \( x_t, u_t, ..., u_{T-1}, I_t, ..., I_{T-1} \):

\[
F_t(x_t, ..., x_{T-1}, u_t, ..., u_{T-1}) = F_t(x_t, u_t, ..., u_{T-1}, I_t, ..., I_{T-1})
\]

Decision rule \( u_t(x_t) \) at each \( t \) is obtained by finding sequence of controls \( u_t, ..., u_{T-1} \) that maximizes \( F_t(x_t, u_t, ..., u_{T-1}, I_t, ..., I_{T-1}) \) for a given state \( x_t \) and a given set of specified inputs \( I_t, ..., I_{T-1} \).
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 from \( t \) through \( T \):

\[
V_t(x_t) = \max_{u_t, \ldots, u_{T-1}} \left[ F_t(x_t, u_t, \ldots, u_{T-1}, I_t, \ldots, I_{T-1}) \right]
\]

Separate benefit term for Stage \( t+1 \):

\[
V_t(x_t) = \max_{u_t} \left[ f_t(u_t, x_t, x_{t+1}) + \max_{u_{t+1}, \ldots, u_{T-1}} F_{t+1}(x_{t+1}, u_{t+1}, \ldots, u_{T-1}, I_{t+1}, \ldots, I_{T-1}) \right]
\]

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

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

Substitute state equation for \( x_{t+1} \):

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

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 \( u_t(x_t) \).

Example 1: Aqueduct Diversions
Maximize benefits from water diverted from 2 locations along an aqueduct. Here the stages correspond to aqueduct sections rather than time (\( T = 2 \)).

State equation:
\[
x_{t+1} = x_t - u_t \quad t = 0, 1, 2
\]

Here stage numbers refer to diversion index rather than time. Benefit at each stage depends only on control (diversion), not on state. Terminal benefit depends on system outflow. No input included in state equation.
Total benefit from diverted water and outflow $x_3$ is:
\[ F(x_1, u_0, u_1) = f_0(u_0) + f_1(u_1) + V_2(x_2) \]

The stage benefit and terminal benefit are:
\[ f_0(u_0) = \frac{1}{2} u_0 \quad f_1(u_1) = u_1 \quad V_2(x_2) = x_2(1-x_2) \]

Additional constraints are:
\[ u_0 \geq 0 \quad u_1 \geq 0 \quad 0 \leq x_2 \leq 1 \]

Solve problem by proceeding backward, starting with last stage ($t = 2$):

**Stage 2**: Find $V_1(x_1)$ for specified $x_1$:
\[
V_1(x_1) = \max_{u_1} \left[ f_1(u_1) + V_2(x_2) \right]
\]
\[
= \max_{u_1} \left[ f_1(u_1) + V_2(x_1 - u_1) \right]
\]
\[
= \max_{u_1} \left[ u_1 + (x_1 - u_1)(1-x_1 + u_1) \right]
\]
\[
= \max_{u_1} \left[ -u_1^2 + 2x_1u_1 + x_1(1-x_1) \right]
\]

Solution that satisfies constraints:
\[ u_1 = x_1 \quad V_1(x_1) = x_1 \]

**Stage 1**: Maximize $V_0(x_0)$ for specified $x_0$:
\[
V_0(x_0) = \max_{u_0} \left[ f_0(u_0) + V_1(x_1) \right]
\]
\[
= \max_{u_0} \left[ \frac{1}{2} u_0 + x_0 - u_0 \right]
\]

Solution that satisfies constraints:
\[ u_0 = 0 \quad V_0(x_0) = x_0 \]

Solution specifies that all water is withdrawn from second diversion point, where it is most valuable.

**Discretization of Variables**
In more complex problems the states, controls and inputs are frequently discretized into a finite number of compatible levels.

Simplest case:
\[ x_t, u_t, I_t \rightarrow x_t^k, u_t^j, I_t^l ; \quad j, k, l = 1, 2, \ldots, L \text{ where } L = \text{number of discrete levels.} \]
Then the optimization of problem at each stage reduces to an exhaustive search through all feasible levels of $u^j_t$ to find the one that maximizes:

$$f_t[u^j_t, x^k_t, g_t(x^k_t, u^j_t, I^l_t)] + V_{t+1}[g_t(x^k_t, u^j_t, I^l_t)]$$

for each feasible $x^k_t$ and a single specified input level $I^l_t$.

This is discrete dynamic programming.

**Example 2: 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:

$$x_{t+1} = x_t - u_t + I_t \quad t = 0, 1, 2$$

Total benefit from released water and final storage $x_3$:

$$F(x_0, u_0, \ldots, u_2) = f_0(u_0) + f_1(u_1) + f_2(u_2) + V_3(x_3)$$

Discretize all variables into compatible levels:

$$u_t = \{0, 1, 2\} \quad x_t = \{0, 1, 2\} \quad I_t = \{0, 1\} \quad t = 0, 1, 2$$

Inflows: $I_0$ $I_1$ $I_2$

1 0 1

Terminal (outflow) benefits: $V_3(x_3) = 0$ for all $x_3$ values

Benefits for each release:

$u_t$ $f_0(u_0)$ $f_1(u_1)$ $f_2(u_2)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Possible state transitions are derived from state equation, inputs, and permissible variable values: Benefit is shown in parentheses after each feasible control value.
Show the possible transitions with a diagram where each feasible state level $x^k_t$ is a circle and each feasible control level $u^j_t$ is a line connecting circles:

```
Stage 1
Stage 2
Stage 3
```

Solve series of 3 optimization problems defined by recursion equation for $t = 2, 1, 0$. Start at last stage and move backward:

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

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

Identify optimum $u_2(x_2)$ values for each possible $x_2$, $V_3(x_3)$ specified as an input:

<table>
<thead>
<tr>
<th>$X_2$</th>
<th>$u_2(x_2)$</th>
<th>$f_2(u_2) + V_3(x_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 + 0 = 0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 + 0 = 1 = $V_2(x_2)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 + 0 = 0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 + 0 = 1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3 + 0 = 3 = $V_2(x_2)$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1 + 0 = 1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3 + 0 = 3 = $V_2(x_2)$</td>
</tr>
</tbody>
</table>
**Stage 2:** Find $V_1(x_1)$ for each level of $x_1$:

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

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$u_1(x_1)$</th>
<th>$f_1(u_1) + V_2(x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 + 1 = 1 $\rightarrow$ Optimum</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 + 3 = 3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4 + 1 = 5 $\rightarrow$ Optimum</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0 + 3 = 4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4 + 3 = 7 $\rightarrow$ Optimum</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5 + 1 = 6</td>
</tr>
</tbody>
</table>

**Stage 1:** Find $V_0(x_0)$ for each level of $x_0$:

$$V_0(x_0) = \max_{u_0} \left[ f_0(u_0) + V_1(x_1) \right] = \max_{u_0} \left[ f_0(u_0) + V_1(x_0 - u_0 + I_0) \right]$$

Identify optimum $u_0(x_0)$ values for each $x_0$, obtain $V_1(x_1)$ from Stage 1:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$u_0(x_0)$</th>
<th>$f_0(u_0) + V_1(x_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 + 5 = 5 $\rightarrow$ Optimum</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3 + 1 = 4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 + 7 = 7 $\rightarrow$ Optimum</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3 + 5 = 8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2 + 1 = 3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3 + 7 = 10 $\rightarrow$ Optimum</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2 + 5 = 7</td>
</tr>
</tbody>
</table>

The optimum $u_t(x_t)$ decision rules for $t = 0, 1, 2$ define a complete optimum decision strategy:

<table>
<thead>
<tr>
<th>$x_0$</th>
<th>$u_0$</th>
<th>$x_1$</th>
<th>$u_1$</th>
<th>$x_2$</th>
<th>$u_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
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 \( V_0(x_0) \).

**Computational Effort**

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: \( 10^3 \)