Chapter 2  Dynamic Programming and Optimal Control of Discrete-Time Systems

2.1  Closed Loop Optimization

Example: Inventory Control

We consider the minimization of the expected cost of ordering quantities in order to meet a stochastic demand.

Rule: The ordering is only possible at discrete time instants \( t_0 < t_1 < \cdots < t_{N-1}, \ N \in \mathbb{N} \).

Excess demand is backlogged and filled as additional inventory becomes available.

\[
\begin{align*}
    x_k & : \text{available stock} \\
    u_k & : \text{order} \quad \left\{ \begin{array}{l}
    \text{at time } t_k, \ 0 \leq k \leq N-1 \\
    \text{at time } t_k, \ 0 \leq k \leq N-1
    \end{array} \right. \\
    w_k & : \text{demand}
\end{align*}
\]

Hence, the stock evolves according to the discrete-time system

\[
x_{k+1} = x_k + u_k - w_k, \quad 0 \leq k \leq N-1.
\]
Costs at time $t_k$:

$r(x_k)$ holding cost (excess inventory) or shortage cost (unsatisfied demand)

$c$ purchasing cost (cost per ordered unit)

Terminal cost:

$R(x_N)$ left over inventory

Total cost:

$E(R(x_N) + \sum_{k=0}^{N-1} (r(x_k) + cu_k))$. 
Consider the minimization problem

\[
\text{(DTS)} \quad \min_{\pi \in \Pi} J_{\pi}(x_0), \quad J_{\pi}(x_0) := E(g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k))
\]

subject to \( x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \ 0 \leq k \leq N - 1 \)

- \( g_k : S_k \times C_k \times D_k \rightarrow \mathbb{R}, \ 0 \leq k \leq N - 1 \) cost functionals
- \( g_N : S_N \rightarrow \mathbb{R} \) terminal cost
- \( f_k : S_k \times C_k \times D_k \rightarrow S_{k+1}, \ 0 \leq k \leq N - 1 \) discrete dynamics
- \( S_k, \ 0 \leq k \leq N \) state spaces \( x_k \in S_k \) states
- \( C_k, \ 0 \leq k \leq N - 1 \) control spaces \( u_k \in \mathcal{U}_k(x_k) \subset C_k \) controls
- \( D_k, \ 0 \leq k \leq N \) disturbance spaces \( w_k \in D_k \) disturbances (random)
- \( \pi = \{\mu_0, \cdots, \mu_{N-1}\} \) control policy
- \( \mu_k : S_k \rightarrow S_k, \ 0 \leq k \leq N - 1 \) control laws
- \( \Pi \) set of admissible control policies
- \( J^*(x_0) = \min_{\pi \in \Pi} J_{\pi}(x_0) \) optimal cost function (optimal value function)

If there exists \( \pi^* \in \Pi \) such that \( J_{\pi^*}(x_0) = J^*(x_0) \), then \( \pi^* \) is called an optimal policy.
Open-loop minimization  (ordering decisions are made at time $t_0$)

\[
\text{minimize } E(R(x_N) + \sum_{k=0}^{N-1} (r(x_k) + cu_k)) \\
\text{subject to } x_{k+1} = x_k + u_k - w_k, \ 0 \leq k \leq N - 1
\]

Closed-loop minimization  (ordering decisions are made at time $t_k$)

Determine a control policy $\pi = \{\mu_k\}_{k=0}^{N-1}$, $\mu_k = \mu_k(x_k)$ such that

\[
\text{minimize } J_\pi(x_0) = E(R(x_N) + \sum_{k=0}^{N-1} (r(x_k) + c\mu_k(x_k))) \\
\text{subject to } x_{k+1} = x_k + \mu_k(x_k) - w_k, \ 0 \leq k \leq N - 1
\]

Dynamic Programming is about the solution of closed-loop minimization problems.
2.2 Bellman’s Optimality Principle (Backward Dynamic Programming)

We consider the minimization subproblems

\[(DTS)_k \min_{\pi_k \in \Pi_k} J_{\pi_k}(x_0), \quad J_{\pi_k}(x_0) := \mathbb{E}(g_N(x_N) + \sum_{\ell=k}^{N-1} g_{\ell}(x_{\ell}, \mu_{\ell}(x_{\ell}), w_{\ell})),\]

subject to \(x_{\ell+1} = f_{\ell}(x_{\ell}, \mu_{\ell}(x_{\ell}), w_{\ell}), \quad k \leq \ell \leq N - 1,\)

where \(\pi_k = \{\mu_k, \ldots, \mu_{N-1}\}\) and \(\Pi_k\) is the set of admissible control policies obtained from \(\Pi\) by deleting all admissible control laws for \(0 \leq \ell \leq k - 1.\)

Bellman’s optimality principle says:

If \(\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}\) is an optimal policy for \((DTS)\), then the truncated policy \(\pi_k^* = \{\mu_k^*, \ldots, \mu_{N-1}^*\}\) is an optimal policy for \((DTS)_k.\)
Bellman’s principle  \implies \text{Solve } (DTS)_k \text{ backwards in time } \implies \text{Backward DP (BDP) algorithm}

\begin{align*}
(BDP)_N & \quad J_N(x_N) = g_N(x_N) \\
(BDP)_k & \quad J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k}(g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))), \quad k = N - 1, N - 2, \ldots, 0
\end{align*}

Theorem (Optimality of the BDP algorithm)

Assumptions:

- The disturbance spaces $D_k$, $0 \leq k \leq N - 1$, are finite or countable sets.
- The expectations in $(BDP)_k$ are finite for every admissible policy.

Then it holds

\begin{equation}
(*) \quad J_k^*(x_k) = J_k(x_k), \quad 0 \leq k \leq N.
\end{equation}

If there exist optimal policies $\mu_k^*$, $0 \leq k \leq N - 1$, for $(BDP)_k$, such that $u_k^* = \mu_k^*(x_k)$, then $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is an optimal policy for $(DTS)$. 
Proof. For $k = N$, (*) holds true by definition of the BDP algorithm. For $k < N$ and $\varepsilon > 0$, we define $\mu^\varepsilon_k(x_k) \in U_k(x_k)$ as the $\varepsilon$-suboptimal control satisfying

\[
(+)_1 \quad E_{w_k}(g_k(x_k, \mu^\varepsilon_k(x_k), w_k) + J_{k+1}(f_k(x_k, \mu^\varepsilon_k(x_k), w_k))) \leq J_k(x_k) + \varepsilon
\]

and define $J^\varepsilon_k(x_k)$ as the expected cost at $x_k$ with respect to the $\varepsilon$-suboptimal policy $\pi^\varepsilon_k = \{\mu^\varepsilon_1(x_k), \ldots, \mu^\varepsilon_{N-1}(x_{N-1})\}$:

\[
(+)_2 \quad J^\varepsilon_k(x_k) = E_{w_k}(g_k(x_k, \mu^\varepsilon_k(x_k), w_k) + J^\varepsilon_{k+1}(f_k(x_k, \mu^\varepsilon_k(x_k), w_k)))
\]

We show by induction on $N - k, k \geq 1$, that

\[
(o)_1 \quad J_k(x_k) = J^\varepsilon_k(x_k) \leq J_k(x_k) + (N - k)\varepsilon,
\]

\[
(o)_2 \quad J^*_k(x_k) = J^\varepsilon_k(x_k) \leq J^*_k(x_k) + (N - k)\varepsilon,
\]

\[
(o)_3 \quad J^*_k(x_k) = J^*_k(x_k).
\]

(i) Begin of induction: $k = N - 1$:

\[
(+) \implies (o)_1 \text{ and } (o)_2 \text{ hold true for } k = N - 1.
\]

Consider $\varepsilon \to 0$ in $(o)_1$ and $(o)_2 \implies (o)_3$. 
(ii) Induction hypothesis: \((o)_1 - (o)_3\) hold true for some \(k + 1\).

(iii) End of induction: Show that \((o)_1 - (o)_3\) hold true for \(k\).

Using \(J^\varepsilon_{k+1}(x_{k+1}) \leq J_{k+1}(x_{k+1}) + (N - k - 1)\varepsilon\) in \((+)\), we find

\[
(\bullet)_1 \quad J^\varepsilon_k(x_k) \leq E_{w_k}(g_k(x_k, \mu^\varepsilon_k(x_k), w_k) + J_{k+1}(f_k(x_k, \mu^\varepsilon_k(x_k), w_k))) + (N - k - 1)\varepsilon
\leq J_k(x_k) + \varepsilon + (N - k - 1)\varepsilon = J_k(x_k) + (N - k)\varepsilon.
\]

On the other hand, using \(J^\varepsilon_{k+1}(x_{k+1}) \geq J_{k+1}(x_{k+1})\) in \((+)\), we obtain

\[
(\bullet)_2 \quad J^\varepsilon_k(x_k) \geq E_{w_k}(g_k(x_k, \mu^\varepsilon_k(x_k), w_k) + J_{k+1}(f_k(x_k, \mu^\varepsilon_k(x_k), w_k)))
\geq \min_{u_k \in U_k(x_k)} E_{w_k}(g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))) = J_k(x_k).
\]

\((\bullet)_1\) and \((\bullet)_2\) \(\implies\) \((o)_1\) for \(k\).

Using $J_{k+1}^\epsilon(x_{k+1}) \leq J_{k+1}(x_{k+1}) + (N - k - 1)\varepsilon$ again in $(+)_2$, yields

\[ (\dagger)_1 \quad J_k^\epsilon(x_k) \leq E_{w_k}(g_k(x_k, \mu_k^\epsilon(x_k), w_k)) + J_{k+1}(f_k(x_k, \mu_k^\epsilon(x_k), w_k)) + (N - k - 1)\varepsilon \]

\[ \leq \min_{u_k \in U_k(x_k)} E_{w_k}(g_k(x_k, u_k, w_k)) + J_{k+1}(f_k(x_k, u_k, w_k)) + \varepsilon + (N - k - 1)\varepsilon \]

\[ \leq \min_{u_k \in U_k(x_k)} E_{w_k}(g_k(x_k, u_k, w_k)) + J_{\pi_k+1}(f_k(x_k, u_k, w_k)) + (N - k)\varepsilon \]

\[ = J_{\pi_k}(x_k) + (N - k)\varepsilon. \]

Minimizing over $\pi_k$: $J_k^\epsilon(x_k) \leq J_k^\epsilon(x_k) + (N - k)\varepsilon$

On the other hand: $J_k^\epsilon(x_k) \leq J_k^\tilde{\epsilon}(x_k)$

\[ \Rightarrow \quad (o)_2 \] for $k$.

Finally, $\varepsilon \to 0$ in $(o)_1, (o)_2 \quad \Rightarrow \quad (o)_3$. 
Example: Inventory Control

Rules:  
- states $x_k \in \{0, 1, 2\}$, controls $u_k \in \{0, 1, 2\}$, demands $w_k \in \{0, 1, 2\}$ with $p(w_k = 0) = 0.1$, $p(w_k = 1) = 0.7$, and $p(w_k = 2) = 0.2$.

- Excess demand $w_k - x_k - u_k$ is lost: $x_{k+1} = \max(0, x_k + u_k - w_k)$.

- Upper bound 2 for stock that can be stored: $x_k + u_k \leq 2$.

- Holding and terminal costs: $r(x_k) = (x_k + u_k - w_k)^2$, $R(x_N) = 0$.

- Ordering cost: $c = 1$.

It follows that

$$g_k(x_k, u_k, w_k) = u_k + (x_k + u_k - w_k)^2, \ 0 \leq k \leq N - 1, \ g_N(x_N) = 0.$$
With $N = 3$ and $x_0 = 0$ the BDP algorithm reads as follows:

$$J_3(x_3) = 0,$$

$$J_k(x_k) = \min_{0 \leq u_k \leq 2 - x_k} E_{w_k}(u_k + (x_k + u_k - w_k)^2 + J_{k+1}(\max(0, x_k + u_k - w_k))), \ 0 \leq k \leq 2.$$ 

**Period 2:** Compute $J_2(x_2)$ for each possible value of $x_2$.

(i) $x_2 = 0$:

$$J_2(0) = \min_{0 \leq u_2 \leq 2} E_{w_2}(u_2 + (u_2 - w_2)^2 + J_3(x_3)) = 0$$

$$u_2 = 0 \implies E_{w_2}(\cdots) = 0.7 \cdot 1 + 0.2 \cdot 4 = 1.5$$

$$u_2 = 1 \implies E_{w_2}(\cdots) = 1 + 0.1 \cdot 1 + 0.2 \cdot 1 = 1.3$$

$$u_2 = 2 \implies E_{w_2}(\cdots) = 2 + 0.1 \cdot 4 + 0.7 \cdot 1 = 3.1$$

It follows that

$$J_2(0) = 1.3, \quad \mu_2^*(0) = 1.$$
(ii) \( x_2 = 1 \): \( x_2 + u_2 \leq 2 \implies 0 \leq u_2 \leq 1 \)

\[
J_2(1) = \min_{0 \leq u_2 \leq 1} E_{w_2}(u_2 + (1 + u_2 - w_2)^2 + J_3(x_3)) = 0
\]

\[
= \min_{0 \leq u_2 \leq 1} (u_2 + 0.1(1 + u_2)^2 + 0.7u_2^2 + 0.2(u_2 - 1)^2)
\]

\[
u_2 = 0 \implies E_{w_2}(\cdots) = 0.1 \cdot 1 + 0.2 \cdot 1 = 0.3
\]

\[
u_2 = 1 \implies E_{w_2}(\cdots) = 1 + 0.1 \cdot 4 + 0.7 \cdot 1 = 2.1
\]

It follows that

\[
J_2(1) = 0.3, \quad \mu_2^*(1) = 0.
\]

(iii) \( x_2 = 2 \): \( x_2 + u_2 \leq 2 \implies u_2 = 0 \)

\[
J_2(2) = E_{w_2}((2 - w_2)^2 + J_3(x_3)) = 0
\]

\[
= 0.1 \cdot 4 + 0.7 \cdot 1 = 1.1
\]

It follows that

\[
J_2(2) = 1.1, \quad \mu_2^*(2) = 0.
\]
Period 2: Bookkeeping  $J_2(0) = 1.3$,  $J_2(1) = 0.3$,  $J_2(2) = 1.1$

Compute  $J_1(x_1)$ for each possible value of  $x_1$.

(i) $x_1 = 0$ :  

$J_1(0) = \min_{0 \leq u_1 \leq 2} E_{w_1}(u_1 + (u_1 - w_1)^2 + J_2(\max(0, u_1 - w_1)))$

$u_1 = 0 \implies E_{w_1}(w_1^2 + J_2(0)) = 0.1 \cdot 1.3 + 0.7 \cdot 2.3 + 0.2 \cdot 5.3 = 2.8$

$u_2 = 1 \implies E_{w_2}(1 + (1 - w_1)^2 + J_2(\max(0, 1 - w_1))) = 1 + 0.1 \cdot (1 + J_2(1)) + 0.7 \cdot J_2(0) + 0.2 \cdot (1 + J_2(0)) = 2.5$

$u_2 = 2 \implies E_{w_2}(2 + (2 - w_1)^2 + J_2(\max(0, 2 - w_1))) = 2 + 0.1 \cdot (4 + J_2(2)) + 0.7 \cdot (1 + J_2(1)) + 0.2 \cdot J_2(0) = 3.68$

It follows that

$J_1(0) = 2.5$,  $\mu_1^*(0) = 1.$
Period 1: Bookkeeping \( J_2(0) = 1.3, \ J_2(1) = 0.3, \ J_2(2) = 1.1 \)

(ii) \( x_1 = 1 \):
\[
J_1(1) = \min_{0 \leq u_1 \leq 1} \mathbb{E}_{w_1}(u_1 + (1 + u_1 - w_1)^2 + J_2(\max(0, 1 + u_1 - w_1)))
\]
\[
u_1 = 0 \implies \mathbb{E}_{w_1}(1 - w_1)^2 + J_2(\max(0, 1 - w_1)) = 0.1 \cdot (1 + J_2(1)) + 0.7 \cdot (0 + J_2(0)) + 0.2 \cdot (1 + J_2(0)) = 1.5
\]
\[
u_1 = 1 \implies \mathbb{E}_{w_1}(1 + (2 - w_1)^2 + J_2(\max(0, 1 - w_1))) = 1 + 0.1 \cdot (4 + J_2(2)) + 0.7 \cdot (1 + J_2(1)) + 0.2 \cdot (0 + J_2(0)) = 2.68
\]

It follows that
\[
J_1(1) = 1.5, \quad \mu_{1}^{*}(1) = 0.
\]

(iii) \( x_1 = 2 \):
\[
J_1(2) = \mathbb{E}_{w_1}((2 - w_1)^2 + J_2(\max(0, 1 - w_1))) = 0.1 \cdot (4 + J_2(2)) + 0.7 \cdot (1 + J_2(1)) + 0.2 \cdot (0 + J_2(0)) = 1.68
\]

It follows that
\[
J_1(2) = 1.68, \quad \mu_{1}^{*}(2) = 0.
\]
Period 0: Bookkeeping $J_1(0) = 2.5$, $J_1(1) = 1.5$, $J_1(2) = 1.68$

(i) $x_0 = 0:$  

$J_0(0) = \min_{0 \leq u_0 \leq 2} E_{w_0}(u_0 + (u_0 - w_0)^2 + J_1(\max(0, u_0 - w_0)))$

$u_0 = 0$  $\implies$  $E_{w_0}(0 - w_0)^2 + J_1(\max(0, 0 - w_0))) = 0.1 \cdot (0 + J_1(0)) + 0.7 \cdot (1 + J_1(0)) + 0.2 \cdot (4 + J_2(0)) = 4.0$

$u_0 = 1$  $\implies$  $E_{w_0}((1 - w_0)^2 + J_1(\max(0, 1 - w_0))) = 1.0 + 0.1 \cdot (1 + J_1(1)) + 0.7 \cdot (0 + J_1(0)) + 0.2 \cdot (1 + J_1(0)) = 3.7$

$u_0 = 2$  $\implies$  $E_{w_0}((2 - w_0)^2 + J_1(\max(0, 2 - w_0))) = 2.0 + 0.1 \cdot (4 + J_1(2)) + 0.7 \cdot (1 + J_1(1)) + 0.2 \cdot (0 + J_1(0)) = 4.818$

It follows that 

$J_0(0) = 3.7$, $\mu^*_0(0) = 1$. 
Solution of the inventory control problem

<table>
<thead>
<tr>
<th>Stock</th>
<th>Period 0 Cost</th>
<th>Period 0 Purchase</th>
<th>Period 1 Cost</th>
<th>Period 1 Purchase</th>
<th>Period 2 Cost</th>
<th>Period 2 Purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.7</td>
<td>1.0</td>
<td>2.5</td>
<td>1.0</td>
<td>1.3</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>2.7</td>
<td>0.0</td>
<td>1.5</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>2.818</td>
<td>0.0</td>
<td>1.68</td>
<td>0.0</td>
<td>1.1</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Example: Copying machine

The Department of Mathematics needs to have a copying machine over the next 5 years. Each new machine costs \( c = 1000 \) $. The maintaining costs for the \( i \)-th year \( 1 \leq i \leq 3 \) are \( m_1 = 60 \) $, \( m_2 = 80 \) $, \( m_3 = 120 \) $. A machine may be kept up to three years before being traded in. The trade-in value is \( s_1 = 800 \) $, \( s_2 = 600 \) $, \( s_3 = 500 \) $. The chairman wants to minimize costs over the 5 year period.

\( 0 \leq t \leq 5 \): year  
\( 0 \leq j \leq 3 \): age of machine  
\( f_t(j) \): minimum cost from time \( 0 \leq t \leq 5 \) to final time 5 given the machine is \( j \) years old at time \( t \)

It follows that

\[
\begin{align*}
f_5(j) &= -s_j \\
f_t(3) &= -s_3 + c + m_1 + f_{t+1}(1) \\
f_t(j) &= \min(-s_j + c + m_1 + f_{t+1}(1), m_{j+1} + f_{t+1}(j+1)), \ 1 \leq j \leq 2 \\
f_0(0) &= c + m_1 + f_1(1)
\end{align*}
\]
Period 5:
\[ f_5(3) = -500, \quad f_5(2) = -600, \quad f_5(1) = -800 \]

Period 4:
\[ f_4(3) = -s_3 + c + m_1 + f_5(1) = -500 + 1000 + 60 - 800 = -240 \quad \text{trade} \]
\[ f_4(2) = \min(-s_2 + c + m_1 + f_5(1), m_3 + f_5(3)) = \]
\[ \min(-600 + 1000 + 60 - 800, 120 - 500) = -380 \quad \text{keep} \]
\[ f_4(1) = \min(-s_1 + c + m_1 + f_5(1), m_2 + f_5(2)) = \]
\[ \min(-800 + 1000 + 60 - 800, 80 - 800) = -540 \quad \text{trade} \]

Period 3:
\[ f_3(3) = -s_3 + c + m_1 + f_4(1) = -500 + 1000 + 60 - 540 = 20 \quad \text{trade} \]
\[ f_3(2) = \min(-s_2 + c + m_1 + f_4(1), m_3 + f_4(3)) = \]
\[ \min(-600 + 1000 + 60 - 540, 120 - 240) = -120 \quad \text{keep} \]
\[ f_3(1) = \min(-s_1 + c + m_1 + f_4(1), m_2 + f_4(2)) = \]
\[ \min(-800 + 1000 + 60 - 540, 80 - 380) = -300 \quad \text{keep} \]
Period 2:
\[ f_3(3) = 20, \quad f_3(2) = -120, \quad f_3(1) = -300 \]
\[ f_2(2) = \min(-s_2 + c + m_1 + f_3(1), m_3 + f_3(3)) = \min(-600 + 1000 + 60 - 300, 120 + 20) = 140 \quad \text{keep} \]
\[ f_2(1) = \min(-s_1 + c + m_1 + f_3(1), m_2 + f_3(2)) = \min(-88 + 1000 + 60 - 300, 80 - 120) = -40 \quad \text{keep or trade} \]

Period 1:
\[ f_1(1) = \min(-s_1 + c + m_1 + f_2(1), m_2 + f_2(2)) = \min(-800 + 1000 + 60 - 540, 80 + 140) = 220 \quad \text{keep or trade} \]

Period 0:
\[ f_0(0) = c + m_1 + f_1(1) = 1000 + 60 + 220 = 1280 \]
Example: Optimal chess match strategy

Rules: \( N \)- game match, \( 0 \leq k \leq N \)
- If a game has a winner, the winner gets 1 point, the loser gets 0 points. In case of a draw, both players get 0.5 points.
- If the score is \( N/2 - N/2 \) at the end of the \( N \) games, the match goes into sudden death.

Playing strategies of player A:
- **Timid play** (\( T_k \)): A draws with probability \( p_d > 0 \) and loses with probability \( 1 - p_d \).
- **Bold play** (\( B_k \)): A wins with probability \( p_w < p_d \) and loses with probability \( 1 - p_w \).

  Timid play never wins, bold play never draws.

State \( x_k \):
Difference points player A - points player B (net score).
Objective: Find policy that maximizes player’s A probability to win the match

Solution by the backward DP algorithm:

Stage N:  \( J_N(x_N) = \begin{cases} 
1 & \text{if } x_N > 0 \\
p_w & \text{if } x_N = 0 \\
0 & \text{if } x_N < 0 
\end{cases} \)

DP recursive scheme:

\[
J_k(x_k) = \max_{T_k:B_k} \left( p_d \ J_{k+1}(x_k) + (1-p_d) \ J_{k+1}(x_k - 1) \right) \\
p_w \ J_{k+1}(x_k + 1) + (1-p_w) \ J_{k+1}(x_k - 1)
\]

- **draw**: net score does not change
- **lose**: net score drops by 1
- **win**: net score increases by 1
- **lose**: net score drops by 1

Optimal to play bold, if

\[
p_w (J_{k+1}(x_k + 1) - J_{k+1}(x_k - 1)) \geq p_d (J_{k+1}(x_k) - J_{k+1}(x_k - 1)) \iff \\
p_w \frac{(J_{k+1}(x_k) - J_{k+1}(x_k - 1))}{J_{k+1}(x_k + 1) - J_{k+1}(x_k - 1)} \geq \frac{p_w}{p_d}
\]
Stage N-1: Set $k = N - 1$

(i) $x_{N-1} > 1 \implies J_{N-1}(x_{N-1}) = \max(p_d J_N(x_{N-1}) + (1 - p_d) J_N(x_{N-1} - 1),$

\[ = \frac{1}{1-p} J_N(x_{N-1} + 1) + (1 - \frac{1}{1-p}) J_N(x_{N-1} - 1) = 1 \]

Optimal play: either

(ii) $x_{N-1} = 1 \implies J_{N-1}(1) = \max(p_d J_N(1) + (1 - p_d) J_N(0),$

\[ = \frac{1}{1-p} J_N(2) + (1 - \frac{1}{1-p}) J_N(0) = p_d + (1 - p_d) p_w \]

Optimal play: timid

(iii) $x_{N-1} = 0 \implies J_{N-1}(0) = \max(p_d J_N(0) + (1 - p_d) J_N(-1),$

\[ = \frac{1}{1-p} J_N(1) + (1 - \frac{1}{1-p}) J_N(-1) = p_w \]

Optimal play: bold
(iv) \[ x_{N-1} = -1 \implies J_{N-1}(-1) = \max(p_d J_N(-1) + (1 - p_d) J_N(-2), \]
\[ p_w J_N(0) + (1 - p_w) J_N(-2) = p_w^2 \]

Optimal play: \textbf{bold}

(v) \[ x_{N-1} < -1 \implies J_{N-1}(x_{N-1}) = 0 \]

Optimal play: \textit{either}

**Conclusion:** The optimal strategy for A is to play timid if and only if A is ahead in the score.
2.3 Finite-State Systems and Shortest Path Problems

Deterministic finite-state system

\[
(FSS) \quad x_{k+1} = f_k(x_k, u_k), \quad 0 \leq k \leq N - 1, \\
u_k = \mu_k(x_k).
\]

- State spaces \( S_k, \ 1 \leq k \leq N \) finite set with \( \text{card}(S_k) = n_k \)
- Controls \( u_k, \ 0 \leq k \leq N - 1 \) provides transition from \( x_k \) to \( f_k(x_k, u_k) \) at cost \( g_k(x_k, u_k) \)
- \( N+1 \) stages \( 0 \leq k \leq N \) with initial stage \( k = 0 \) and final stage \( k = N \)
Description of (FSS) by a graph

- Initial node $s_0$
- Nodes $s_k$, $1 \leq k \leq n_k$, at stage $k$
- Terminal node $s_{N+1}$
- $a_{ij}^k$: transition cost for transition from state $x_i \in S_k$ to state $x_j \in S_{k+1}$
- $a_{ij}^k = \infty$, if there is no control providing a transition from $x_i \in S_k$ to $x_j \in S_{k+1}$
- $a_{is_{N+1}}^N$: terminal cost for transition from state $x_i \in S_N$ to state $s_{N+1}$
Backward DP algorithm

\begin{align*}
J_N(s_i) &= a_{iN+1}^N, \quad 1 \leq i \leq N \\
J_k(s_i) &= \min_{1 \leq j \leq n_{k+1}} (a_{ij}^k + J_{k+1}(s_j)), \quad 1 \leq i \leq n_k, \quad 0 \leq k \leq N - 1
\end{align*}

Remarks:
\begin{itemize}
  \item $J_k(s_i)$: optimal-cost-to-go from state $x_i \in S_k$ to terminal state $s_{N+1}$
  \item $J_0(s_0)$: optimal cost
\end{itemize}

Shortest path problem:
Identify transition costs $a_{ij}^k$ with lengths of arcs connecting nodes.
Optimal cost corresponds to the shortest path from the initial node $s_0$ to the terminal node $s_{N+1}$. 
Shortest path problem as deterministic finite-state system

Consider a graph with \( N + 1 \) nodes and arcs connecting node \( i \) with node \( j \) of length \( a_{ij} \).
The final node \( N + 1 \) is called the destination.

Rules:
- Total of \( N \) moves (but degenerate moves from \( i \) to \( i \) at cost \( a_{ii} = 0 \) are allowed).
- \( a_{ij} = +\infty \), if there is no arc connecting \( i \) with \( j \).

Formulation as deterministic finite-state system:
\[
J_k(i) : \text{optimal-cost-to-go from node } i \text{ to } N + 1 \text{ in } N - k \text{ moves.}
\]

\[
\begin{align*}
\text{(BDP)} & \quad J_{N-1}(i) = a_{i,N+1}, \quad 1 \leq i \leq N \quad \text{(one move)} \\
J_k(i) & = \min_{1 \leq j \leq N} (a_{ij} + J_{k+1}(j)), \quad 1 \leq i \leq N, \quad k = N - 2, \ldots, 0
\end{align*}
\]
Example: Shortest path problem (N=4)

1 move:
\[ J_3(4) = 3, \quad J_3(3) = 5, \quad J_3(2) = 7, \quad J_3(1) = 2 \]

2 moves:
\[ J_2(4) = \min_{1 \leq j \leq 4} (a_{4j} + J_3(j)) = \min(2 + 2, 5 + 7, 1 + 5, 0 + 3) = 3 \]
\[ J_2(3) = \min_{1 \leq j \leq 4} (a_{3j} + J_3(j)) = \min(5 + 2, 0.5 + 7, 0 + 5, 1 + 3) = 4 \]
\[ J_2(2) = \min_{1 \leq j \leq 4} (a_{2j} + J_3(j)) = \min(6 + 2, 0 + 7, 0.5 + 5, 5 + 3) = 5.5 \]
\[ J_2(1) = \min_{1 \leq j \leq 4} (a_{1j} + J_3(j)) = \min(0 + 2, 6 + 7, 5 + 5, 2 + 3) = 2 \]
The Forward DP algorithm

Applicable in case of deterministic shortest path problems, since an optimal path from $s_0$ to the terminal state $s_{N+1}$ is also optimal for the reverse shortest path problem: Find the shortest path from $s_{N+1}$ to $s_0$ with reverse order of arcs but same lengths.

\[(FDP) \quad \tilde{J}_N(s_j) = a_{s_0j}^0, \quad 1 \leq j \leq n_1 \]

\[\tilde{J}_k(s_j) = \min_{1 \leq i \leq a_{N-k}} (a_{ij}^{N-k} + \tilde{J}_{k+1}(s_i)), \quad 1 \leq j \leq n_{N-k+1}, \quad 1 \leq k \leq N - 1 \]

Optimal cost:

\[\tilde{J}_0(s_{N+1}) = \min_{1 \leq i \leq n_N} (a_{i,s_{N+1}}^N + \tilde{J}_1(s_i)).\]
Optimal State Estimation for Partially Observable Finite-State Markov Chains

**Definition:** A matrix $P = (p_{ij})_{i,j=1}^N \in \mathbb{R}^{N \times N}$ is said to be a (right) stochastic matrix, if

$$p_{ij} \geq 0, \quad 1 \leq i, j \leq N, \quad \sum_{j=1}^{N} p_{ij} = 1, \quad 1 \leq i \leq N.$$ 

**Definition:** A pair $\{S, P\}$ is called a stationary finite-state Markov chain, if $S = \{1, \cdots, N\}$ is a finite number of states, $P = (p_{ij})_{i,j=1}^N \in \mathbb{R}^{N \times N}$ is a (right) stochastic matrix, and there is a process with initial state $x_0 \in S$ with probability

$$P(x_0 = i) = r^i_0, \quad 1 \leq i \leq N,$$

and possible transitions from state $x_k$ to state $x_{k+1}, 0 \leq k \leq N - 1$, with probability

$$P(x_{k+1} = j; x_k = i) = p_{ij}, \quad 1 \leq i, j \leq N.$$
It follows that

\[ P(x_k = j; x_0 = i) = (P^k)_{ij}, \quad 1 \leq i, j \leq N. \]

**Probability distribution** of state \( x_k \) after \( k \) transitions:

\[
\begin{align*}
    r_k &= (r_k^1, \ldots, r_k^N) \in \mathbb{R}^N, \\
    r_k^j &= \sum_{i=1}^{N} P(x_k = j; x_0 = i)r_0^i = \sum_{i=1}^{N} (P^k)_{ij}r_0^i, \quad 1 \leq j \leq N.
\end{align*}
\]

**Partially observable finite-state Markov chain (hidden Markov model)**

**Feature:** States in transition \( x_{k-1} \to x_k \) unknown, but observation \( z_k \to x_k \) of probability \( r(z_k; x_{k-1}, x_k) \) available.

**State transition sequence:** \( X_N = \{x_0, x_1, \ldots, x_N\} \)

**Observation sequence:** \( Z_N = \{z_1, z_2, \ldots, z_N\} \)

**Conditional probability:**

\[
p(X_N \mid Z_N) = \frac{p(X_N, Z_N)}{p(Z_N)}
\]
Optimal State Estimation

Find \( \hat{X}_N = \{\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_N\} \) such that
\[
p(\hat{X}_N \mid Z_N) = \max_{X_N} p(X_N \mid Z_N).
\]

Since \( p(Z) \) is constant (once \( Z_N \) is known):
\[
(*) \quad p(\hat{X}_N, Z_N) = \max_{X_N} p(X_N, Z_N).
\]

Lemma (Probability of state transition/observation sequences)

Let \( X_N, Z_N \) be the state transition and observation sequences of a partially observable finite-state Markov chain \( \{S,P\} \) and let \( r_{x_0} \) and \( r(z_k; x_{k-1}, x_k) \) be the probabilities that the initial state is \( x_0 \) and of the observations \( z_k \) for the \( k\)-th state transition \( x_{k-1} \rightarrow x_k \). Then, it holds
\[
p(X_N, Z_N) = r_{x_0} \prod_{k=1}^{N} p_{x_{k-1}, x_k} r(z_k; x_{k-1}, x_k).
\]

Proof. Induction on \( N \) (homework).
Optimal State Estimation as a Shortest Path Problem

Trellis diagram: Concatenate \( N + 1 \) copies of the state spaces at stages \( 0 \leq k \leq N \) and introduce initial node \( s_0 \) and terminal node \( s_{N+1} \).

An arc connects a node \( x_{k-1} \) at stage \( k-1 \) with a node \( x_k \) at stage \( k \), if for the associated transition probability it holds \( p_{x_{k-1},x_k} > 0 \).

Lemma (Optimal state estimation as a minimization problem)

The maximization problem \((*)\) is equivalent to

\[
\text{(o)} \quad \min_{X_N} \left( -\ln(r_{x_0}) - \sum_{k=1}^{N} \ln(p_{x_{k-1},x_k} \cdot r(z_k; x_{k-1}, x_k)) \right).
\]
Corollary (Optimal state estimation as a shortest path problem)

The solution of the minimization problem \( o \) is given by the shortest path in the Trellis diagram connecting \( s_0 \) and \( s_{N+1} \).

Proof. Assign lengths of the arcs according to

\[
\begin{align*}
(s_0, x_0) &\mapsto -\ln(r_{x_0}), \\
(x_{k-1}, x_k) &\mapsto -\ln(p_{x_{k-1}, x_k} r(z_k; x_{k-1}, x_k)), \quad 1 \leq k \leq N, \\
(x_N, s_{N+1}) &\mapsto 0.
\end{align*}
\]

Viterbi Algorithm (Forward DP Algorithm)

\[
\begin{align*}
\text{dist}(s_0, x_0) &= -\ln(r_{x_0}), \\
\text{dist}(s_0, x_{k+1}) &= \min_{x_k : p_{x_k, x_{k+1}} > 0} (\text{dist}(s_0, x_k) - \ln(p_{x_{k+1}, x_{k+1}} r(z_{k+1}; x_k, x_{k+1}))), \quad 0 \leq k \leq N.
\end{align*}
\]
Example: Speech Recognition: Convolutional Coding and Decoding

An important issue in information theory is the reliable transmission of binary data over a noisy communication channel, e.g., transmitting speech data from the cell phone of a sender via satellite to the cell phone of a receiver. This can be done by convolutional coding and decoding.

**Binary data sequence:** \( W_N = \{w_1, w_2, \cdots, w_N\}, \ w_k \in \{0, 1\}, \ 1 \leq k \leq N, \)

**Coded sequence:** \( Y_N = \{y_1, y_2, \cdots, y_N\}, \ y_k = (y_{1k}, \cdots, y_{nk})^T \in \mathbb{R}^n, \ y_k \in \{0, 1\}, \ 1 \leq i \leq n, \ 1 \leq k \leq n, \)

**Received sequence:** \( Z_N = \{z_1, z_2, \cdots, z_N\}, \)

**Decoded sequence:** \( \hat{W}_N = \{\hat{w}_1, \hat{w}_2, \cdots, \hat{w}_N\}. \)

**Goal:** \( \hat{W}_N \) should be as close to \( W_N \) as possible.
Convolutional Coding

Realization: Discrete-time dynamical system

\[
\begin{align*}
(DTS) \quad & y_k = Cx_{k-1} + dw_k, \quad 1 \leq k \leq N, \\
& x_k = Ax_{k-1} + bw_k, \quad 1 \leq k \leq N,
\end{align*}
\]

where \( A \in \mathbb{R}^{m \times m}, \ C \in \mathbb{R}^{n \times n}, \ b \in \mathbb{R}^m, \ d \in \mathbb{R}^n, \ x_0 \in \mathbb{R}^m, \) are given.

Remark: Products and sums are evaluated using modulo 2 arithmetic.

Example: \( N = 4, \ m = 2, \ n = 3 \)

\[
\begin{align*}
C &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, & A &= \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \\
b &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, & d &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.
\end{align*}
\]

In the figure, \( 0/000 \) etc. means Data/Codework \( w_k/y_k \).
Explanation of the state transition diagram

a) \( x^k \) = \( \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \) \( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \) + \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) = \( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \) = \( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \) = \( x^k \),

\( y^k \) = \( \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \) \( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \) + \( \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \) = \( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \) = \( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \) = \( y^k \)

b) \( x^k \) = \( \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \) \( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \) + \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) = \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) = \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) = \( x^k \),

\( y^k \) = \( \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \) \( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \) + \( \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \) = \( \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \) = \( \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \) = \( y^k \)
Example: \( N = 4, m = 2, n = 3, x_0 = 00 \)

\[
\{w_1, w_2, w_3, w_4\} = \{1, 0, 0, 1\} \rightarrow \begin{cases} \text{State sequence} & \{x_1, x_2, x_3, x_4\} = \{01, 11, 10, 00\} \\ \text{Codeword sequence} & \{y_1, y_2, y_3, y_4\} = \{111, 011, 111, 011\} \end{cases}
\]

Transmission over noisy channel:
Codewords \( y_k \) are received as \( z_k \) with known conditional probability \( P(z_k \mid y_k) \).

\[
P(Z_N \mid Y_N) = \prod_{k=1}^{N} P(z_k \mid y_k)
\]

Maximum likelihood estimation:
Find \( \hat{Y}_N = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N) \) such that

\[
\text{(MLE)} \quad P(Z_N \mid \hat{Y}_N) = \max_{Y_N} P(Z_N \mid Y_N)
\]

\( Y_N \) satisfies (DTS)
Reformulation as shortest path problem via the Trellis diagram

- Dummy initial node $s$, dummy terminal node $t$,
- $s$ connected to $t$ by zero-length arc,
- $-\ln P(z_k | y_k)$ length of arc associated with codeword $y_k$.

\[
\begin{align*}
\text{minimize} \quad & -\sum_{k=1}^{N} \ln(P(z_k | y_k)) \quad \text{over } Y_N \\
\text{subject to} \quad & Y_N \text{ satisfies (DTS)}
\end{align*}
\]

Viterbi algorithm:

\[
d_{k+1}(x_{k+1}) = \min_{x_k} \left( d_k(x_k) - \ln(P(z_{k+1} | y_{k+1})) \right)
\]

$x_k$ is connected to $x_{k+1}$ by an arc $(x_k, x_{k+1})$ with codeword $y_{k+1}$

Optimal decoded sequence:

$\hat{W}_N$ retrieved from $\hat{Y}_N$ by the Trellis diagram (look up optimal pairs $\hat{w}_k | \hat{y}_k$).
Label Correcting Methods

Consider a graph with initial node \( s \), terminal node \( t \), intermediate nodes \( i \), and arcs \( \text{arc}(i,j) \) of length \( a_{ij} \geq 0 \) connecting node \( i \) and node \( j \).

A node \( j \) is said to be the child of a node \( i \), if there exists \( \text{arc}(i,j) \) with \( a_{ij} \geq 0 \).

Features: Iterative detection of shorter paths from node \( s \) to every other node \( i \) in the graph.

- **Label:** The label \( d_i \) is a variable whose value corresponds to the current shortest path from \( s \) to \( i \) within the iterative process. The label \( d_t \) of the terminal node \( t \) is stored in a variable called \( \text{UPPER} \).

- **Candidate list:** The candidate list, called \( \text{OPEN} \) is an (ordered) list of nodes containing candidates (nodes) for possible inclusion in the shortest path.
Initialization of labels and of OPEN

- Set $d_s = 0$ and $d_i = +\infty$, $i \neq s$.
- OPEN = $\{s\}$ (i.e., initially OPEN only contains the initial node).

k-th step of the iterative process

Substep 1: Node removal

Remove a node $i$ from OPEN according to a certain strategy, e.g., the Bellman-Ford method (First in/first out): Remove node from top of OPEN.

Substep 2: Label reduction and update of OPEN/UPPER

For each child $j$ of $i$ check

(*) $d_i + a_{ij} < \min(d_j, \text{UPPER})$

If (*) is satisfied, there is a shorter path from $i$ to $j$ then previously detected. Set

(+) $d_j = d_i + a_{ij}$

and call node $i$ the parent of node $j$. 
Update of OPEN/UPPER

OPEN = OPEN ∪ \{j\}, if j \neq t; Place j at the top of OPEN
UPPER = d_i + a_{it}, if j = t

Substep 3: Termination criterion

If OPEN = \emptyset, terminate the algorithm.
Otherwise, set k = k + 1 and go to substep 1.
Traveling Salesman Problem

A salesman who lives in city A has to visit N − 1 other cities for business before returning home. Find the minimum mileage trip that starting from A visits each other city once and then returns to A.

Example:
Traveling Salesman Problem

Initialization: \( d_1 = 0, \ d_i = +\infty, \ 2 \leq i \leq 10, \ \text{OPEN} = \{1\}, \ \text{UPPER} = \{\infty\} \)

Iteration 1: Remove 1 from OPEN \( \implies \) OPEN = \( \emptyset \)

Nodes 2, 7, 10 are the children of node 1

\( d_1 + a_{1j} < \min(d_j, \text{UPPER}), \ j \in \{2, 7, 10\} \)

\( d_2 = \{a_{12}\} = \{5\}, \ d_7 = \{a_{17}\} = \{1\}, \ d_{10} = \{a_{1,10}\} = \{15\} \)

OPEN = \{2, 7, 10\}, UPPER = \{\infty\}

Iteration 2: Remove 2 from OPEN \( \implies \) OPEN = \{7, 10\}

Nodes 3, 5 are the children of node 2

\( d_2 + a_{2j} < \min(d_j, \text{UPPER}), \ j \in \{3, 5\} \)

\( d_3 = \{d_2 + a_{23}\} = \{25\}, \ d_5 = \{d_2 + a_{25}\} = \{9\} \)

OPEN = \{3, 5, 7, 10\}, UPPER = \{\infty\}
Traveling Salesman Problem

Iteration 3: Remove 3 from OPEN \implies OPEN = \{5, 7, 10\}

Node 4 is the child of node 3
\[ d_4 + a_{34} < \min(d_4, \text{UPPER}) \]
\[ d_4 = \{d_3 + a_{34}\} = \{28\} \]
OPEN = \{4, 5, 7, 10\}, UPPER = \{\infty\}

Iteration 4: Remove 4 from OPEN \implies OPEN = \{5, 7, 10\}

Node t is the child of node 4
\[ d_t = \{d_4 + a_{4t}\} = \{43\} \]
OPEN = \{5, 7, 10\}, UPPER = \{43\}

Iteration 5: Remove 5 from OPEN \implies OPEN = \{7, 10\}

Node 6 is the child of node 5
\[ d_5 + a_{56} < \min(d_6, \text{UPPER}) \]
\[ d_6 = \{d_5 + a_{56}\} = \{12\} \]
OPEN = \{6, 7, 10\}, UPPER = \{43\}
Traveling Salesman Problem

Iteration 6: Remove 6 from OPEN \(\implies\) OPEN = \{7, 10\}

Node \(t\) is the child of node 6
\[d_t = \{d_6 + a_{6t}\} = \{13\}\]
OPEN = \{7, 10\}, UPPER = \{13\}

Homework: Continue with the iteration

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Node existing OPEN</th>
<th>OPEN at the end of the iteration</th>
<th>UPPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>–</td>
<td>1</td>
<td>\infty</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2,7,10</td>
<td>\infty</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3,5,7,10</td>
<td>\infty</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4,5,7,10</td>
<td>\infty</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5,7,10</td>
<td>43</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6,7,10</td>
<td>43</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7,10</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8,10</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9,10</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>\emptyset</td>
<td>13</td>
</tr>
</tbody>
</table>
Branch-and-Bound Methods

Given a finite set $X$ of feasible points $x \in X$, $\text{card } X = n$, and a function $f : X \rightarrow \mathbb{R}$ consider the minimization problem

\[(*) \quad \min_{x \in X} f(x)\]

Features:

- Partition of the feasible set $X$ into subsets with associated minimization subproblems.
- Computation of bounds for the optimal value of the minimization subproblems and exclusion of subsets.

Computation of bounds:

Let $Y_1, Y_2 \subset X$ such that

$$f_1 \leq \min_{x \in Y_1} f(x), \quad \bar{f}_2 \geq \min_{x \in Y_2} f(x)$$

Exclusion principle:

If $\bar{f}_2 \leq f_1$, exclude $Y_1$, since the associated costs can not be smaller than the cost associated with the optimal solution in $Y_2$. 
Systematic Implementation of Branch-and-Bound Methods

Let $\mathcal{P}_X$ be the collection of subsets of $X$. Construct an acyclic graph whose nodes are elements of $\mathcal{P}_X$ according to the following rules:

- $X \in \mathcal{P}_X$.
- For each $x \in X$: $\{x\} \in \mathcal{P}_X$.
- Each set $Y \in \mathcal{P}_X$ is partitioned into subsets $Y_i \in \mathcal{P}_X$, $1 \leq i \leq k$, such that
  \[ Y = \bigcup_{i=1}^{k} Y_i, \quad Y_i \neq Y, \quad 1 \leq i \leq k. \]

The set $Y$ is called the parent of $Y_i$, $1 \leq i \leq k$, whereas the sets $Y_i$ are said to be the children of $Y$.

- Each set $Z \in \mathcal{P}_X$, $Z \neq X$, has at least one parent.
- The arcs are those connecting parents and children.
Example:

Specification of the lengths \( \text{arc}(Y, Y_i) \):

For every parental node \( Y \) compute upper and lower bounds:

\[
\begin{align*}
    f_Y &\leq \min_{x \in Y} f(x) \leq \bar{f}_Y, \\
    \text{arc}(Y, Y_i) &= f_{Y_i} - f_Y.
\end{align*}
\]

The solution of \( (\ast) \) corresponds to the shortest path from the initial node \( X \) to one of the nodes \( x \).
Branch-and-Bound Method as Label Correcting Method

Initialization: \( \text{OPEN} = \{X\}, \text{UPPER} = \bar{f}_X \)

Substep 1: Remove a node \( Y \) from \( \text{OPEN} \)

Substep 2: For each child \( Y_j \) of \( Y \) check

\[ (o) \quad f_{Y_j} < \text{UPPER}. \]

If \((o)\) is satisfied:

\[
\text{OPEN} = \text{OPEN} \cup \{Y_j\}, \quad \text{UPPER} = \bar{f}_{Y_j}, \quad \text{if } \bar{f}_{Y_j} < \text{UPPER}.
\]

If \( Y_j \) consists of a single feasible point, mark the solution as the best solution so far.

Substep 3: If \( \text{OPEN} = \emptyset \), terminate. The best solution so far is an optimal solution.

Otherwise, continue the iteration with Substep 1.