Dynamic Programming

Dynamics:
• $x_{k+1} = f_k(x_k, u_k, w_k),\ k = 0, 1, ..., N-1$
where $x_k \in S_k, u_k \in U_k(x_k)$, and $w_k \backsim p_{w_k|x_k,u_k}$ with $p_{w_k|x_k,u_k,*} = p_{w_k|x_k,u_k}, \forall * \in \{x_l, u_l, w_l | l < k\}$
• admissible policy:$\pi = (\mu_0(.), \mu_1(.), ..., \mu_{N-1}(.))$
$u_k = \mu_k(x_k), u_k \in U_k(x_k), k = 0, 1, ..., N-1$

Expected Cost:
Given $x \in S_0$, the expected closed loop cost of starting at $x_0 = x$ associated with policy π is: $J_\pi(x) = \underset{(X_1, W_0|x_0=x)}{E}[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)]$,
where $X_1 = (x_1, ..., x_N), W_0 = (w_0, ..., w_{N-1})$

Objective:
Construct an optimal policy π^* s.t. $\forall x \in S_0$:

$$\pi^* = \underset{\pi\in\Pi}{argmin} J_\pi(x)$$

*Open loop control can never give better performance than closed loop control (u_k depends on x_k) since open loop control is a special case of closed loop control. In the absence of disturbances w_k, the two give theoretically the same performance.
Consider a system with N_x distinct states and N_u distinct control inputs: There are a total of N_u^N different open loop strategies. There are a total of $N_u(N_u^{N_x})^{N-1}$ different closed loop strategies.

Transition Probability:
$P_{ij}(u, k) = P(x_{k+1} = j | x_k = i, u_k = u) = p_{x_{k+1}|x_k,u_k}(j|i, u) = p_{w_k|x_k,u_k}(j|i, u) = \sum_{\bar{w}_k|f_k(i,u,\bar{w}_k)=j} p_{w_k|x_k,u_k}(\bar{w}_k|i, u)$

Principle of Optimality: Let π^* be an optimal policy. Consider the subproblem whereby we are at $x \in S_i$ at time i and we want to minimize: $\underset{X_{i+1},W_i|x_i=x}{E}[g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)]$
where $X_{i+1} = (x_{i+1}, ..., x_N) and W_i = (w_i, ..., w_{N-1})$.
Then the truncated policy $\pi^* = (\mu_i^*(.), ..., \mu_{N-1}^*(.))$ is optimal for this problem

DPA:
Initialization: $J_N(x) = g_N(x), \forall x \in S_N$
Recursion:
$J_k(x) = \underset{u\in U_k(x)}{min} \underset{(w_k|x_k=x,u_k=u)}{E}[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))], \forall x \in S_k, k = N-1, ..., 0$
*We calculate cost-to-go J_k with expected value, where we don't consider variance $Var(x) = E(x^2) - E(x)^2$
*Computation: $N_u N_x(N-1) + N_u$ operations

Time Lags: Assume the dynamics becomes:

$$x_{k+1} = f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k)$$

Let $y_k = x_{k-1}, s_k = u_{k-1}, \tilde{x}_k = (x_k, y_k, s_k)$
$\tilde{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} f_k(x_k, y_k, u_k, s_k, w_k) \\ x_k \\ u_k \end{bmatrix} = \tilde{f}_k(\tilde{x}_k, u_k, w_k)$

Correlated Disturbances
If $w_k = C_k y_{k+1}, y_{k+1} = A_k y_k + \xi_k$, where $\xi_k, k = 0, ..., N-1$ are independent random variables.
• Let the augmented state vector $\tilde{x}_k = (x_k, y_k)$. Note that now y_k must be observed at time k, which can be done using a state estimator. • $\tilde{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} f_k(x_k, u_k, w_k = C_k(A_k y_k + \xi_k)) \\ A_k y_k + \xi_k \end{bmatrix} = \tilde{f}_k(\tilde{x}_k, u_k, \xi_k)$

Forecast
At the beginning of each period k, we receive a prediction y_k (forecast), we know a collection of distributions $\{p_{w_k|y_k}(.|.), ...\}$ and priori The forecast itself has a given a-priori probability distribution $p(\xi_k)$ with $y_{k+1} = \xi_k$. y_{k+1}: this event happens on day k+1, ξ_k: the forecast about y_{k+1} on day k
• New state vector: $\tilde{x}_k = (x_k, y_k)$, new disturbance: $\tilde{w}_k = (w_k, \xi_k)$
$\tilde{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} f_k(x_k, u_k, w_k) \\ \xi_k \end{bmatrix} = \tilde{f}_k(\tilde{x}_k, u_k, \xi_k)$ •
The dynamics becomes: $J_k(\tilde{x}) = \underset{u\in U_k(x)}{min} \underset{(w_k|y_k=y)}{E}[g_k(x, u, w_k) + \underset{\xi_k}{E}[J_{k+1}(f_k(x, u, w_k), \xi_k)]]$
$= \underset{u\in U_k(x)}{min} \underset{(w_k|y_k=y)}{E}[g_k(x, u, w_k) + \sum_{i=1}^m p_{\xi_k}(i) J_{k+1}(f_k(x, u, w_k), i)]$
$\forall x \in S_k, y \in \{1, ..., m\}, k = N-1, ..., 0$

Infinite Horizon Problem: as N goes infinity, let $\mathbf{V_l}(.) = \mathbf{J_{N-l}}(.)$, V converges, so we have $J(x) = \underset{u\in U_k(x)}{min} \underset{(w|x=x,u=u)}{E}[g(x, u, w) + J(f(x, u, w))], \forall x \in S$,
i.e. **Bellman Equation** $->$ optimal policy is time invariant.

Stochastic Shortest Path Problem (SSP)
• **Dynamics:**
$x_{k+1} = w_k, P(w_k = j | x_k = i, u_k = u) = P_{ij}(u)$(time-invariant transition probability), $\forall x_k \in S, u \in U(i)$
U, S are finite, • **Cost:**
$J_\pi(i) = \underset{(X_1, W_0|x_0=i)}{E}[\sum_{i=0}^{N-1} g(x_k, \mu_k(x_k), w_k)]$

Assumption 4.1 Cost-free termination state:
State 0 is denoted as the termination state with $S = 0, 1, ..., n$, where
$P_{00}(u) = 1, g(0, u, 0) = 0, \forall u \in U(0)$
A stationary policy μ is said to be **proper** if, when using this policy, there exists an integer m such that: $P(x_m = 0 | x_0 = i) > 0$

Assumption 4.2 proper policy: There exists at least one proper policy $\mu \in \Pi$. Furthermore, for every improper policy μ', the corresponding cost function $J_{\mu'}(i)$ is infinity for at least one state $i \in S$.
*This assumption is required in order to guarantee that a unique solution to the BE exists for the SSP problem, which will then be the optimal cost.
*It ensures that a policy exists for which the probability of reaching the termination state goes to one as the time horizon N goes to infinity. It also ensures that the policies for which this does not occur incur infinite cost, which ensures that there are no non-positive cycles.

Theorem for SSP:*under assumption 4.1 and 4.2,
1.Given any initial conditions $V_0(1), ..., V_0(n)$, the sequence $V_l(i)$:
$V_{l+1}(i) = \underset{u\in U(i)}{min}(q(i, u) + \sum_{j=1}^n P_{ij}(u) V_l(j)), \forall i \in S^+$(1)
where $S^+ = S\backslash 0$ and $q(i, u) = \underset{(w|x=i,u=u)}{E}[g(x, u, w)]$
converges to the optimal cost $J^*(i)$ for all $i \in S^+$
2.The optimal cost satisfy the BE:
$J^*(i) = \underset{u\in U(i)}{min}(q(i, u) + \sum_{j=1}^n P_{ij}(u) J^*(j)), \forall i \in S^+$
3. The solution to the BE is unique
4. The minimizing u for each $i \in S^+$ of the BE gives an optimal policy, which is proper.

Value Iteration:(1) above, until a threshold for $\|V_{l+1}(i) - V_l(i)\|$ is reached

Policy Iteration: • **initialization:** Initialize with a proper policy $\mu^0 \in \Pi$
• **Policy evaluation:** Given a policy μ_h, solve for the corresponding cost J_{μ^h} by solving the linear system
$J_{\mu^h}(i) = q(i, \mu^h(i)) + \sum_{j=1}^n P_{ij}(\mu^h(i)) J_{\mu^h}(j), \forall i \in S^+$
• **Policy Improvement:** Obtain a new stationary policy μ^{h+1}:
$\mu^{h+1}(i) = \underset{u\in U(i)}{argmin}(q(i, u) + \sum_{j=1}^n P_{ij}(u) J_{\mu^h}(j)), \forall i \in S^+$, iterate until $J_{\mu^{h+1}}(i) = J_{\mu^h}(i)$ for all $i \in S^+$

Computational Complexity:
• **PI:** - Stage 1 solves a system of n linear equations in n unknowns, i.e. $O(n^3)$ – Stage 2 involves n minimizations over p possible control inputs, and evaluating the sum takes n steps. Thus, the complexity is $O(n^2 p)$ • **VI:** n minimizations over p possible control inputs, and evaluating the sum takes n steps. Thus, the complexity is $O(n^2 p)$
* At each iteration, PI is more computationally expensive than VI. But theoretically it takes an infinite number of iterations for VI to converge, whereas with PI, in the worst case the number of iterations is p^n.

Other variants of PI and VI:
• *Gauss-Seidel Update:* use new values in VI
For $i = 1$ to n :
$V(i) \leftarrow \underset{u\in U(i)}{min}(q(i, u) + \sum_{j=1}^n P_{ij}(u) V(j))$
•*Asynchronous PI: ???*
• *Connection to Linear Algebra:*
Stage 1 of PI $\rightarrow J = G + PJ \rightarrow (I-P)J = G$
*there exists a unique solution for J if and only if $(I-P)$ is invertible. $(I-P)$ is guaranteed to be invertible when the policy is proper.
*$(I-P)^{-1} = \sum_{k=0}^\infty P^k$

Linear Programming:
$\underset{V}{maximize} \sum_{i\in S^+} V(i)$ s.t.
$V(i) \le (q(i, u) + \sum_{j=1}^n P_{ij}(u) V(j)), \forall u \in U(i), i \in S^+$

Discounted Problem:
We introduce *discount factor* α,
$J_\pi(i) = \underset{(X_1, W_0|x_0=i)}{E}[\sum_{k=0}^{N-1} \alpha^k \tilde{g}(x_k, \mu_k(x_k), w_k)]$

Auxiliary SSP Problem: for discounted problem
• State: $x_k \in S = S^+ \cup \{0\} = \{0, 1, ..., n\}$, 0 is virtual termination state
• Control: $U(x_k) \forall x_k \in S^+$ remains same, $U(0) = stay$.
Policy $\pi = (\mu_0(.), \mu_1(.), ..., \mu_{N-1}(.))$, s.t.
$u_k = \mu_k(x_k), u_k \in U(x_k), \forall x_k \in S$
• Dynamics:
$P_{ij}(u) = \alpha \tilde{P}_{ij}(u), u \in U(i), \forall i, j \in S^+$
$P_{i0}(u) = 1 - \alpha, u \in U(i), \forall i \in S^+$
$P_{0j}(u) = 0, u = stay, \forall j \in S^+$
$P_{00}(u) = 1, u = stay$
• Cost:
$g(x_k, u_k, w_k) = \alpha^{-1} \tilde{g}(x_k, u_k, w_k), \forall u_k \in U(x_k), x_k, w_k \in S^+$
$g(x_k, u_k, 0) = 0, \forall u_k \in U(x_k), x_k \in S^+$
$g(0, stay, 0) = 0$
•$J_\pi(r) = \underset{(X_1, W_0|x_0=i)}{E}[\sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), w_k)]$

The shortest Path (SP) Problem
• Graph: defined by a finite vertex space V and a weighted edge space
$C = \{(i, j, c_{i,j}) \in V \times V \times \mathbb{R} \cup \{\infty\} | i, j \in V\}$ where c_{ij} denotes the arc length or cost.
• Path: A path is defined as an ordered list of nodes, that is $Q = (i_1, i_2, ..., i_q)$
*The set of all paths that start at some node $S \in V$ and end at node $T \in V$ is denoted by $\mathbb{Q}_{S,T}$.
• Objective: Determine $Q^* = \underset{Q\in\mathbb{Q}_{S,T}}{argmin} J_Q$
* Assumption for SP: no negative cycles

Deterministic Finite State (DFS) Problem
(no expected value, no transition probability, no disturbances)
• Dynamics
$x_{k+1} = f_k\{x_k, u_k\}, x_k \in S_k, k = 0, ..., N, u_k \in U_k(x_k), k = 0, ..., N-1$ (no input for k=N)
• Cost function: $g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$

DFS $\rightarrow$ SP:

To be precise, the vertex space is the union of all stage and state pairs, that is:

$$\mathcal{V} := \left(\bigcup_{k=0}^{N} \mathcal{V}_k\right) \cup \{\mathsf{T}\},$$

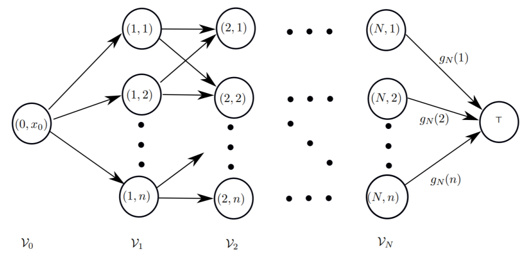
where,

$$\mathcal{V}_0 := \{(0, x_0)\}$$
$$\mathcal{V}_k := \{(k, x_k) | x_k \in \mathcal{S}_k\},\ k = 1, \ldots, N,$$
$$\mathsf{s} := (0, x_0).$$

The weighted edge space is then:

$$\mathcal{C} := \left\{ ((k, x_k), (k+1, x_{k+1}), c) \middle| \begin{array}{l} (k, x_k) \in \mathcal{V}_k \\ (k+1, x_{k+1}) \in \mathcal{V}_{k+1} \\ c = \underset{\{u\in\mathcal{U}_k(x_k)|x_{k+1}=f_k(x_k,u)\}}{\min} g_k(x_k, u) \\ k \in \{0, \ldots, N-1\} \end{array} \right\} \bigcup \{((N, x_N), \mathsf{T}, g_N(x_N)) | (N, x_N) \in \mathcal{V}_N\}.$$

Stage:0,...,N; state:1,...n



SP $\rightarrow$ DFS

- The state space is:
$$\mathcal{S}_k := \mathcal{V}\backslash\{\mathsf{T}\} \text{ for } k = 1, ..., N-1, \mathcal{S}_N := \{\mathsf{T}\} \text{ and } \mathcal{S}_0 := \{\mathsf{s}\}.$$
- The control space is:
$$\mathcal{U}_k := \mathcal{V}\backslash\{\mathsf{T}\} \text{ for } k = 0, ..., N-2, \text{ and } \mathcal{U}_{N-1} := \{\mathsf{T}\}.$$
- The dynamics are:
$$x_{k+1} = u_k, \quad u_k \in \mathcal{U}_k,\ k = 0, \ldots, N-1.$$
- The stage cost functions are:
$$g_k(x_k, u_k) := c_{x_k,u_k}, \quad k = 0, ..., N-1,$$
$$g_N(\mathsf{T}) := 0.$$

We can solve the DFS problem using DPA, where $J_k(i)$ is the optimal cost of getting from node i to node T in $N - k = |\mathcal{V}| - 1 - k$ moves:

$$J_N(\mathsf{T}) = g_N(\mathsf{T}) = 0,$$
$$J_k(i) = \min_{u\in\mathcal{U}_k} (g_k(i, u) + J_{k+1}(u)), \quad \forall i \in \mathcal{S}_k,\ k = N-1, \ldots, 0,$$
$$\Rightarrow J_{N-1}(i) = c_{i,T}, \quad \forall i \in \mathcal{V}\backslash\{\mathsf{T}\},$$
$$J_k(i) = \min_{j\in\mathcal{V}\backslash\{\mathsf{T}\}} (c_{i,j} + J_{k+1}(j)), \quad \forall i \in \mathcal{V}\backslash\{\mathsf{T}\},\ k = N-2, \ldots, 1,$$
$$J_0(\mathsf{s}) = \min_{j\in\mathcal{V}\backslash\{\mathsf{T}\}} (c_{\mathsf{s},j} + J_1(j)).$$

*Remark: We can terminate the algorithm early if $J_k(i) = J_{k+1}(i)$ for all $i \in V\backslash\{T\}$

Hidden Markov Models: •Dynamics:
$x_{k+1} = w_k, x_k \in S,\ P_{ij} = p_{w|x}(j|i), \forall i, j \in S$ the distributions p_{x_0} and $p_{w|x}$ are given (*Markov Chain*)
• Measurement: $M_{ij}(z) = p_{z|x,w}(z|i, j)$ (given, *likelihood function*)

• Objective: Given a measurement sequence $Z_1 = (z_1, ..., z_N)$, find the "most likely" state trajectory $X_0 = (x_0, ..., x_N) = \underset{X_0}{argmax} p(X_0|Z_1)$

Viterbi Algorithm:

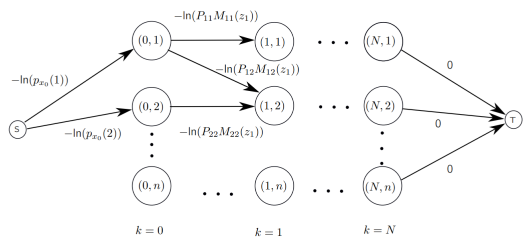
$p(X_0, Z_1) = p(X_0|Z_1)p(Z_1) \rightarrow argmax p(X_0, Z_1)$

$p(X_0, Z_1) = p(x_0) \prod_{k=1}^{N} P_{x_{k-1}x_k} M_{x_{k-1}x_k}(z_k)$

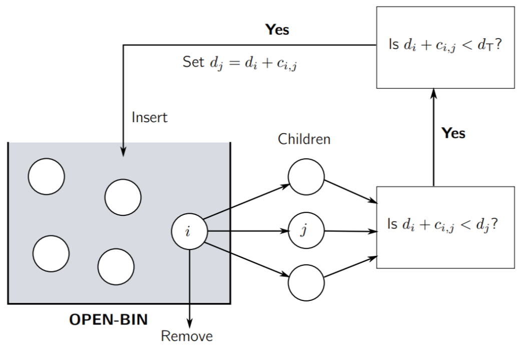
$\implies \underset{X_0}{minimize}(c_{S,(0,x_0)} + \sum_{k=1}^{N} c_{(k-1,x_{k-1}),(k,x_k)})$

$$c_{\mathsf{S},(0,x_0)} = \begin{cases} -\ln(p(x_0)) & \text{if } p(x_0) > 0 \\ \infty & \text{if } p(x_0) = 0 \end{cases},$$

$$c_{(k-1,x_{k-1}),(k,x_k)} = \begin{cases} -\ln(P_{x_{k-1}x_k} M_{x_{k-1}x_k}(z_k)) & \text{if } P_{x_{k-1}x_k} M_{x_{k-1}x_k}(z_k) > 0 \\ \infty & \text{if } P_{x_{k-1}x_k} M_{x_{k-1}x_k}(z_k) = 0 \end{cases}.$$



Label correcting methods:


Yes
Is $d_i + c_{i,j} < d_\mathsf{T}$?
Set $d_j = d_i + c_{i,j}$
Insert
Children
Yes
i
j
Is $d_i + c_{i,j} < d_j$?
OPEN-BIN
Remove

Initialization: Place node S in OPEN, set $d_S = 0, d_j = \infty \forall j \in V \backslash \{S\}$

*Theorem: If there exists at least one finite cost path from S to T, then the LCA terminates with $d_T = J_{Q^*}$. Otherwise, the LCA termimnates with ∞.

• **Depth-First Search** or "last in, first out" strategy, that is, a node is always removed from the top of the OPEN bin and each node entering OPEN is placed at the top. It is often implemented as a stack.

• **Breadth-First Search** or "first in, first out" strategy; that is, a node is always removed from the top of the OPEN bin and each node entering OPEN is placed at the bottom. It is often implemented as a queue.

• **Best-First Search (Dijkstra's Algorithm)**: At each iteration, the node that is removed from OPEN is the node i^* where $d_{i^*} = \underset{i \in OPEN}{min} d_i$, i.e. remove the node that currently has the best label.

• A^***-algorithm:**in step 2 of LCA, we strengthen requirement of a node j being admitted to OPEN from $d_i + c_{i,j} < d_T$ to $d_i + c_{i,j} + h_j < d_T$, where h_j is some positive lower bound on the cost to get from node j to T (heuristic).