Approximate Dynamic Programming

- Recall our general DP formulation for problems with disturbances:

$$J^*(x_0) = \min_{\pi \in \Pi} \mathbb{E}_w \Big[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \Big]$$

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \forall k \in \{0, 1, ..., N-1\}$$

- And the (backwards) DP algorithm:

$$J_N(x_n) = g_N(x_N)$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \Big\{ \mathbb{E}_{w_k} \Big[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \Big] \Big\}, \forall k \in \{0, ..., N-1\}$$

Approximate Dynamic Programming

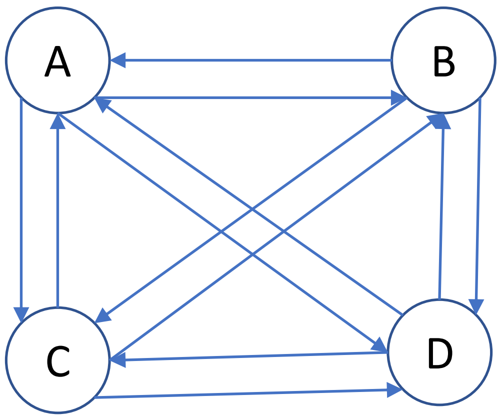
- Our goal, as always, is to compute the optimal closed-loop policies:

$$\pi^* = \{\mu_0^*(x_0), ..., \mu_{N-1}^*(x_{N-1})\}$$

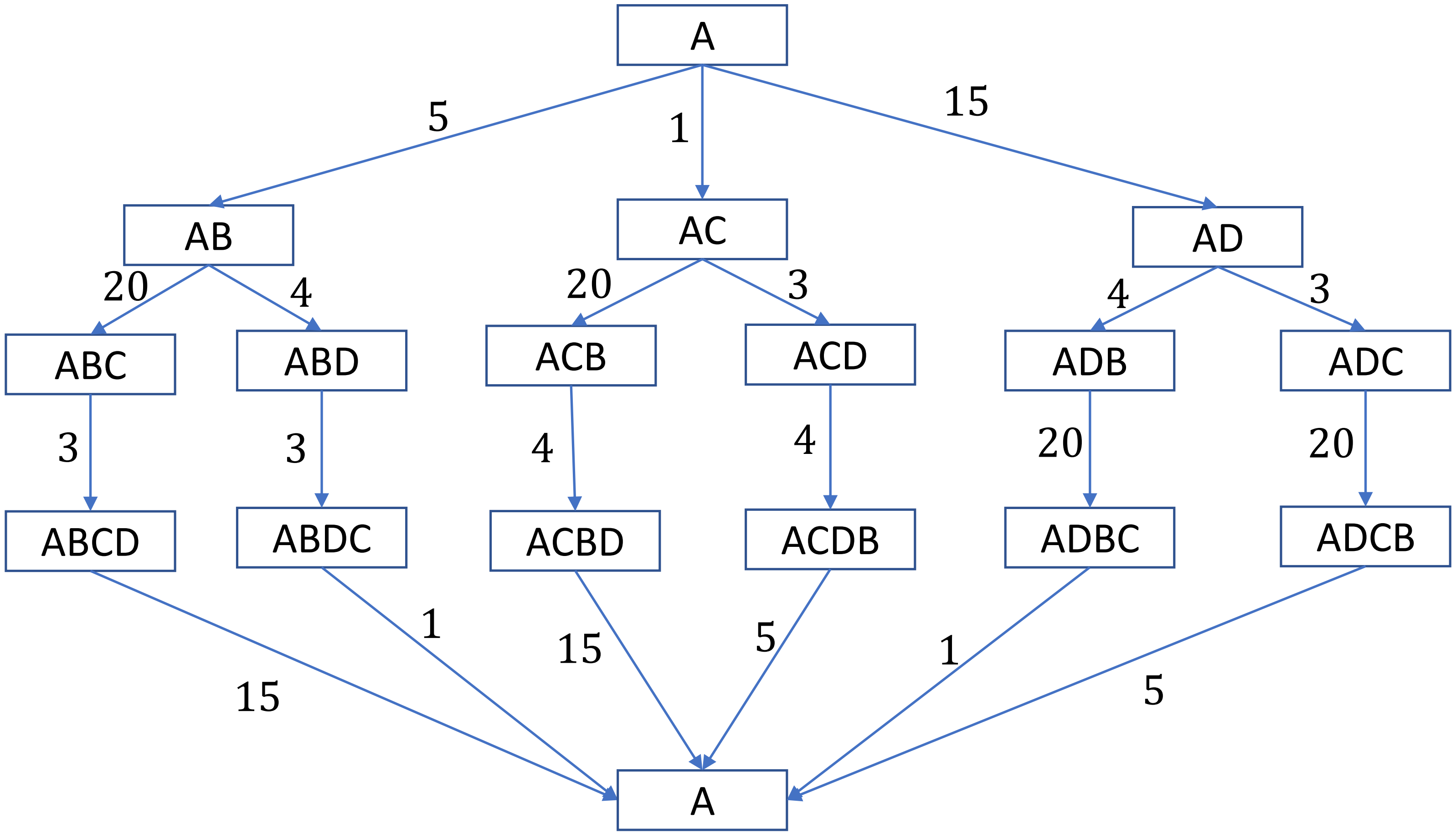
- The key challenges we observed were:
  - How to compute the expectation w.r.t. to w_i?
  - How to perform the optimization on the right-hand side?
  - How to overcome the fact the above has to be done for **every** possible state?

- But how hard are these challenges?

Example: Traveling Salesman Problem



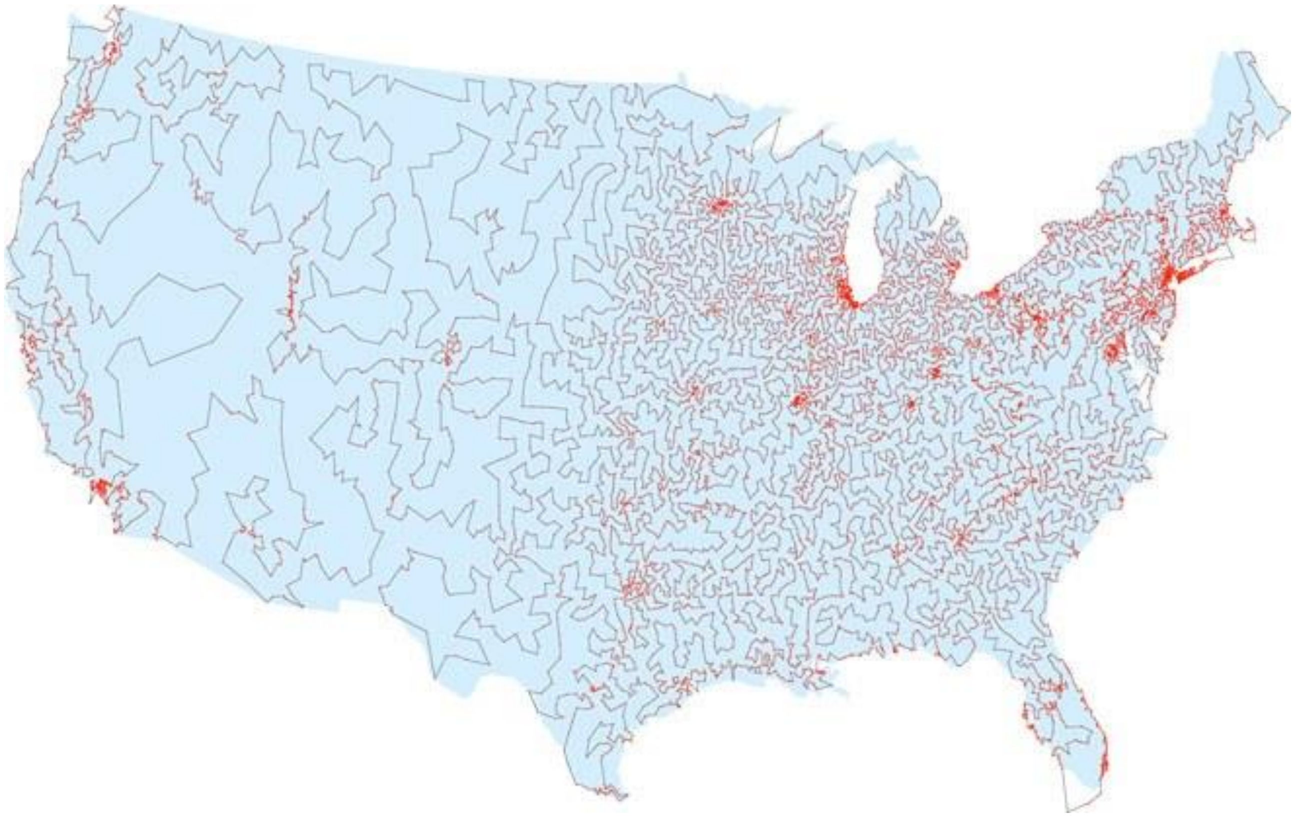
Distances: $\begin{bmatrix} 0 & 5 & 1 & 15 \\ 5 & 0 & 20 & 4 \\ 1 & 20 & 0 & 3 \\ 15 & 4 & 3 & \end{bmatrix}$



- The number of stages in the DP is exponential with the number of nodes!

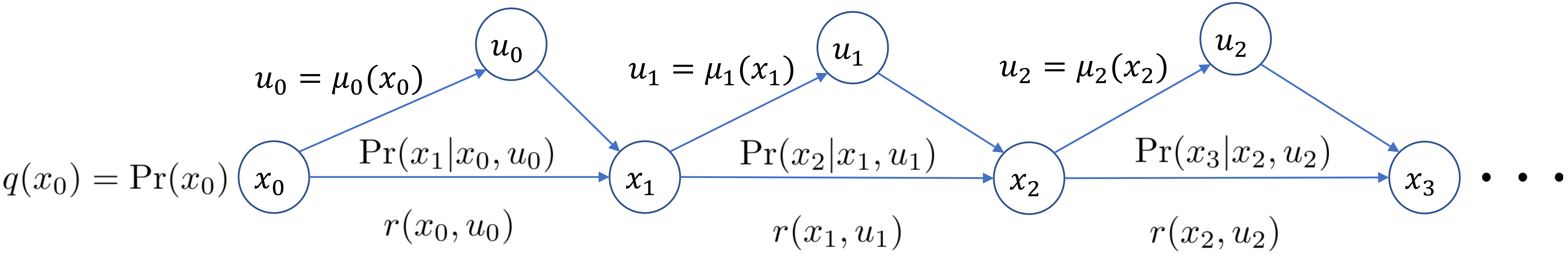
Example: Traveling Salesman Problem

- If the curse of dimensionality is present and the number of stages, even in the deterministic case explodes with the problem size, then how this is achieved?



Markov Decision Processes (MPD)

- Before we address approximation in DP, let's recall the MDP formulation:



- Where we highlight the direct relation between MDP and the DP framework:

$$x_{k+1} = w_k \qquad r(x_k, u_k) = -g_k(x_k, u_k, w_k)$$

$$w_k \sim \Pr(w_k|x_k, u_k)$$

- What elements can be approximated in the graphical model above?

Q-factor reformulation

- Consider the DP recursion:

$$J_N(x_n) = g_N(x_N)$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \left\{ \mathbb{E}_{w_k} \left[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right] \right\}, \forall k \in \{0, \ldots, N-1\}$$

- Let's define the **Q-function (or Q-factors)**:

$$Q_k^*(x_k, u_k) = \mathbb{E}_{w_k} \left[g_k(x_k, u_k, w_k) + J_{k+1}^*(f_k(x_k, u_k, w_k)) \right], \forall k \in \{0, \ldots, N-1\}$$

- Where $J_{k+1}^*(x_{k+1})$ are the optimal cost-to-go functions for each stage *k.* Then we can write:

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \left\{ Q_k^*(x_k, u_k) \right\}, \forall k \in \{0, \ldots, N-1\}$$

Approximation in Value Space

- In addition, we re-write the DP recursion in terms of the Q-factors:

$$Q_k^*(x_k, u_k) = \mathbb{E}_{w_k}\left[g_k(x_k, u_k, w_k) + \min_{u_{k+1} \in U_{k+1}(f_k(x_k, u_k, w_k))} \left\{Q_{k+1}^*(f_k(x_k, u_k, w_k), u_{k+1})\right\}\right]$$

$$\forall k \in \{0, \dots, N-1\}$$

- Suppose we had a function $\tilde{J}_{k+1}(x_{k+1})$ that approximates the cost-to-go for each stage k. And for each stage we compute the following minimization:

$$\tilde{\mu}_k(x_k) = \arg\min_{u_k \in U_k(x_k)} \left\{\mathbb{E}_{w_k}\left[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\right]\right\}, \forall k \in \{0, \dots, N-1\}$$

- Note that the policy $\tilde{\pi} = (\tilde{\mu}_0, \dots, \tilde{\mu}_{N-1})$ is admissible and sub-optimal.

Approximation in Value Space

- We can write the same sub-optimal policy in terms of the now approximate Q-factors:

$$\tilde{Q}_k(x_k, u_k) = \mathbb{E}_{w_k}\left[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\right], \forall k \in \{0, \dots, N-1\}$$

$$\tilde{\mu}_k(x_k) = \arg \min_{u_k \in U_k(x_k)} \left\{\tilde{Q}_k(x_k, u_k)\right\}, \forall k \in \{0, \dots, N-1\}$$

- How to obtain a good approximation $\tilde{J}_k(x_k)$ is the central focus of the first family of Reinforcement Learning algorithms we will study: The Value Space approximation methods.

Example: Multistep Lookahead

- As a simple but very important example is the case where $\tilde{J}_{k+1}(x_{k+1})$ is itself given by a one-stage DP recursion:

$$\tilde{J}_{k+1}(x_{k+1}) = \min_{u_{k+1} \in U_{k+1}(x_{k+1})} \Big\{ \mathbb{E}_{w_{k+1}} \Big[g_{k+1}(x_{k+1}, u_{k+1}, w_{k+1}) + \\ \tilde{J}_{k+2}(f_{k+1}(x_{k+1}, u_{k+1}, w_{k+1})) \Big] \Big\}$$

- Where $\tilde{J}_{k+2}(x_{k+2})$ is yet another approximation of the cost-to-go, now from stage 2.

- For the l-step lookahead, $\tilde{J}_{k+1}(x_{k+1})$ is given by:

$$\tilde{J}_{k+1}(x_{k+1}) = \min_{(\mu_{k+1}, \ldots, \mu_{k+l-1})} \mathbb{E}_{w_{k+1}, \ldots, w_{k+l}} \left[\tilde{J}_{k+l}(x_{k+l}) + \sum_{i=k+1}^{k+l-1} g_i(x_i, \mu_i(x_i), w_i) \right]$$

Example: Multistep Lookahead

- For problems with very large horizon (or infinite), we can use a "large-enough" lookahead l to let the final approximation $\tilde{J}_{k+l}(x_{k+l})$ to be very simple (for example equal to zero).

- Recall our last example about doing LQR with a horizon $M \ll N$, where instead of using the limiting algebraic Riccati Equation we solved a sub-problem with truncated horizon:

$$\min_{u_0,\dots,u_{M-1}} \left\{ \mathbb{E}_w \left[x_M^\top Q x_M + \sum_{i=0}^{M-1} (x_i^\top Q x_i + u_i^\top R u_i) \right] \right\}$$

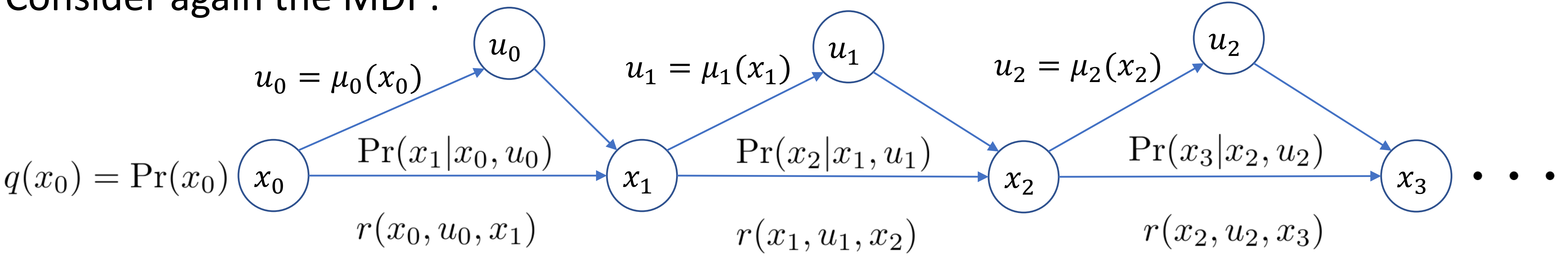
$$x_{i+1} = Ax_i + Bu_i + w_i, \ \forall i \in \{0, \dots, M-1\}$$

- After solving the M-stage discrete Riccati Equation we applied:

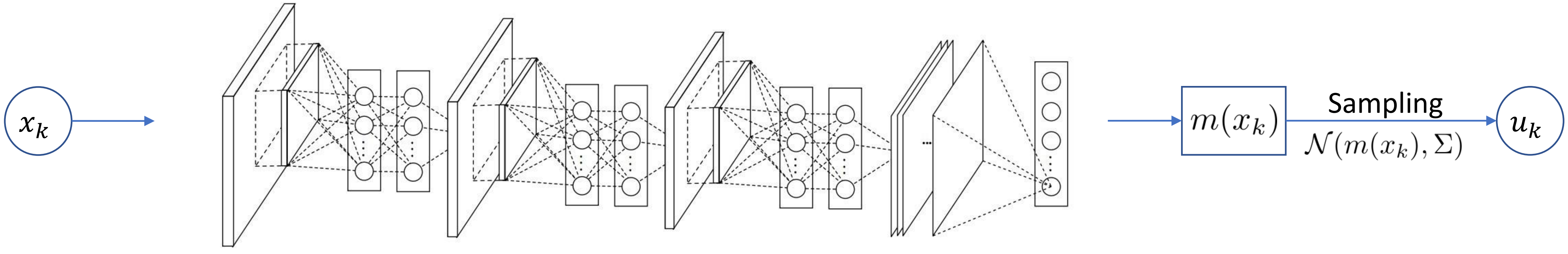
$$\tilde{\mu}_0(x_0) = K_0 x_0$$

Approximation in Policy Space

- Consider again the MDP:



- Suppose we now approximate the policy functions, by some parametric function, like a Neural Network:



$$\tilde{\mu}_k(x_k) \sim \mathcal{N}(m(x_k), \Sigma)$$

Example: Randomized Policy

- Let's define as $\pi_\theta(u_k|x_k)$ the probability distribution of the controls/actions u_k given the state x_k. And let θ be the parameters of the Neural Network.

- Like we did in the HMM case, we can write the probability of a whole trajectory as:

$$\Pr(\underbrace{x_0, u_0, \ldots, x_{N-1}, u_{N-1}, x_N}_{\tau}; \theta) = q(x_0) \prod_{i=0}^{N-1} \Pr(x_{i+1}|x_i, u_i)\pi_\theta(u_i|x_i) = p(\tau;\theta)$$

- Then we can optimize over all possible sequences (**Policy Gradient**):

$$\theta^* = \arg\max_\theta \left\{ \mathbb{E}_{p(\tau;\theta)} \left[\sum_{k=0}^{N-1} r(x_k, u_k) \right] \right\} = \arg\max_\theta \left\{ \sum_{k=0}^{N-1} \mathbb{E}_{p_\theta(x_k,u_k)} \left[r(x_k, u_k) \right] \right\}$$

$$p_\theta(x_{k+1}, u_{k+1}) = \Pr(x_{k+1}|x_k, u_k)\pi_\theta(u_{k+1}|x_{k+1})p(x_k, u_k) \qquad p_\theta(x_0, u_0) = \pi_\theta(u_0|x_0)q(x_0)$$

Model-based X Model-free

- Let's now address the expectation issue:

$$J_N(x_n) = g_N(x_N)$$

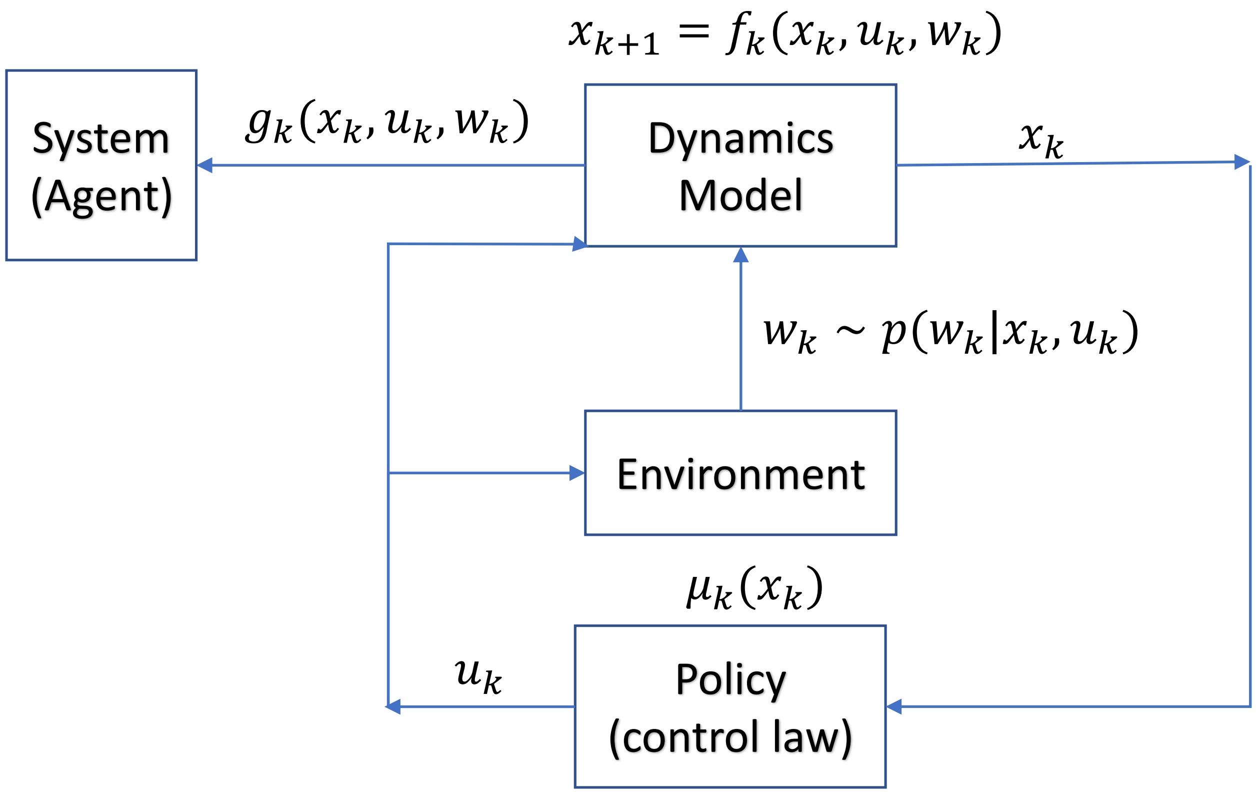
$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \left\{ \mathbb{E}_{w_k} \left[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right] \right\}, \forall i \in \{0, ..., N-1\}$$

- How the probabilities are computed? Do we have the distributions?

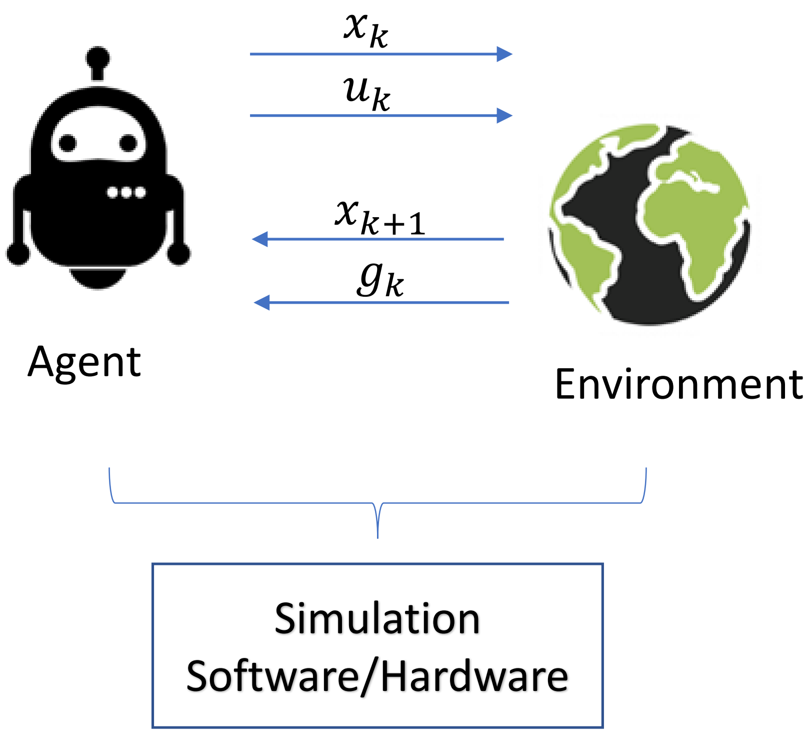
- **Model-based case:** In this case we **know** the distributions in closed-form. That is we have $p(w_k|x_k, u_k)$, for every triplet (x_k, u_k, w_k)**.** Moreover, the functions f_k and g_k are known. Expecations are computed algebraic calculations.

- **Model-free case:** In this case, we need to rely on Monte-Carlo **simulations** to compute expectations. Moreover, we may not the functions f_k and g_k and we also have to rely on simulations to obtain the system transitions and costs/rewards.

Model-based X Model-free



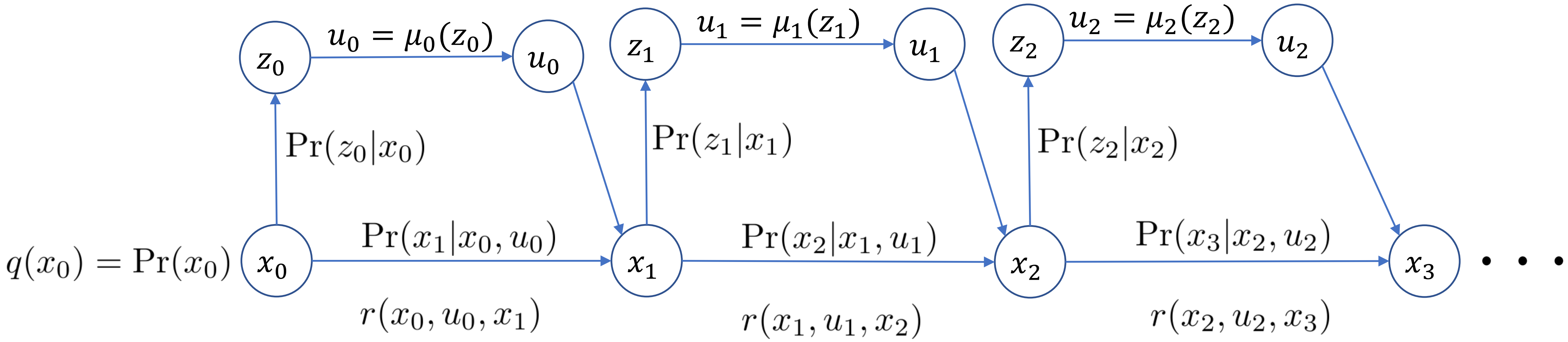
Model-based Case



Model-free Case

Imperfect Information Case: POMPD's

- Like the HMM , most often in practical application we do not have access to perfect state information. Hence the graphical model can be adapted to:



- Notice now that policy $\mu_k(z_k)$ is given the observation z_k and **not** the state x_k!

- Can the policy depend on the whole history? That is:

$$u_k = \mu_k(z_0, z_1, \ldots, z_k, u_0, u_1, \ldots, u_{k-1})$$

DP with Imperfect Information

- We will present here the most general form of the DP formulation where the closed loop policy $\mu_k(\cdot)$ depends on the whole history $I_k = (z_0, z_1, \ldots, z_k, u_0, u_1 \ldots, u_{k-1})$:

$$J^*(I_0) = \min_{\pi \in \Pi} \mathbb{E}_{w,v}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(I_k), w_k)\right]$$

$$x_{k+1} = f_k(x_k, \mu_k(I_k), w_k), \forall k \in \{0, 1, ..., N-1\}$$

$$z_k = h_k(x_k, \mu_{k-1}(I_{k-1}), v_k), \forall k \in \{0, 1, ..., N-1\}$$

$$z_0 = h_0(x_0, v_0), \forall k \in \{0, 1, ..., N-1\}$$

- Notice that the $v_k's$ can be seen as observation noise and we can draw a direct relationship between the DP formulation above and the POMPD's (we leave it as an exercise).

- And we assume that: $v_k \sim \Pr(\cdot | x_{k-1}, u_k, w_k)$

DP with Imperfect Information

- Recall the idea of sufficient statistics, which for the HMM were the counts of transitions. Suppose we are able to find a sufficient statistics function $S_k(I_k)$ for every information vector I_k .

- The intuition is that S_k contains all the *relevant* information about I_k. So we would be able to write the optimal policy as:

$$\mu_k^*(I_k) = \bar{\mu}_k(S_k(I_k)), \forall k \in \{0, ..., N-1\}$$

- For some functions $\bar{\mu}_k's$.

- Like we did on the EM Algorithm, let's consider here the conditional probability of the state x_k given the history I_k(there, this probability could be seen as a **belief**!)

DP with Imperfect Information

- Namely let b_k be the **belief state**: $b_k = \Pr(x_k | I_k)$

- Suppose we had in hand a way of computing the beliefs ("The E-Step"), via some recursive formula:

$$b_{k+1} = \Phi_k(b_k, u_k, z_{k+1})$$

- Then we re-write the (backwards) recursion as a Perfect Information DP:

$$\bar{J}(b_k) = \min_{u_k \in U_k(b_k)} \left\{ \mathbb{E}_{x_k, w_k, z_{k+1}} \left[g_k(x_k, u_k, w_k) + \bar{J}_{k+1}(\Phi(b_k, u_k, z_{k+1})) | I_k, u_k \right] \right\}$$

$$\bar{J}_{N-1}(b_{N-1}) = \min_{u_{N-1} \in U_{N-1}(b_{N-1})} \left\{ \mathbb{E}_{x_{N-1}, w_{N-1}} \left[g_N(f_{N-1}(x_{N-1}, u_{N-1} w_{N-1})) + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) | I_{N-1}, u_{N-1} \right] \right\}$$

DP with Imperfect Information

- And it follows that:

$$J^*(I_0) = E_{z_0}\left[\bar{J}_0(b_0)\right]$$

- Note how nice this formulation is!

- The “states” now are the beliefs b_k. The dynamics are given by the forward recursion:

$$b_{k+1} = \Phi_k(b_k, u_k, z_{k+1})$$

- The controls are the same. Lastly z_k plays the role of the “disturbance”.

- It makes sense, since from stage k we only have knowledge of the history I_k, hence the future observations $(z_{k+1}, \dots z_N)$ are considered in expectation.

DP with Imperfect Information

- This reformulation is called the **Belief MDP reduction** of MOMPD's.
- Lastly, as we run the DP forward, our tasks are decomposed two parts as well (!)
- First, we have the **estimator** part which computed the belief:

$$b_k = \Pr(x_k | I_k)$$

- Given the history I_k gathered so far. Then, we have the **actuator** part which computes:

$$\mu_k^*(I_k) = \bar{\mu}_k(b_k)$$

- This separation leads to yet another family of Approximation Methods, which works on the beliefs, instead of the actual system states.

Other dimensions for approximations

- We saw three main types of approximations that can be done:
  - Approximations in the Value Space
  - Approximations in the Policy Space
  - Approximations in computing expectations (simulations)

- Other aspects of approximations are:
  - Offline X Online methods: Multi-parametric programs and online querying
  - Problem Decomposition: Benders Decomposition, Lagrange Relaxations
  - Aggregation methods: features extraction, state reduction

- There are a **huge** number of algorithms, ideas in all the areas above as this is a very active area of research. We will explore the main algorithms as they often the base for the more sophisticated ideas.