AA 274
Principles of Robotic Autonomy

Decision making and dynamic programming

Basic Decision Making Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k),\ k = 0, \dots, N$

Basic Decision Making Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k),\ k = 0, \ldots, N$
- Control constraints: $u_k \in U(x_k)$

Basic Decision Making Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \ldots, N$
- Control constraints: $u_k \in U(x_k)$
- Probability distribution: $P_k(\cdot|x_k, u_k)$ of w_k

Basic Decision Making Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \ldots, N$
- Control constraints: $u_k \in U(x_k)$
- Probability distribution: $P_k(\cdot|x_k, u_k)$ of w_k
- Policies: $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where $u_k = \mu_k(x_k)$

Basic Decision Making Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \ldots, N$
- Control constraints: $u_k \in U(x_k)$
- Probability distribution: $P_k(\cdot | x_k, u_k)$ of w_k
- Policies: $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where $u_k = \mu_k(x_k)$
- Expected Cost:

$$J_\pi(x_0) = E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

Basic Decision Making Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k), \; k = 0, \ldots, N$
- Control constraints: $u_k \in U(x_k)$
- Probability distribution: $P_k(\cdot | x_k, u_k)$ of w_k
- Policies: $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where $u_k = \mu_k(x_k)$
- Expected Cost:

$$J_\pi(x_0) = E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

- Decision making problem

$$J^*(x_0) = \min_\pi J_\pi(x_0)$$

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal closed-loop policy
- Additive cost (central assumption)
- Risk-neutral formulation

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal closed-loop policy
- Additive cost (central assumption)
- Risk-neutral formulation

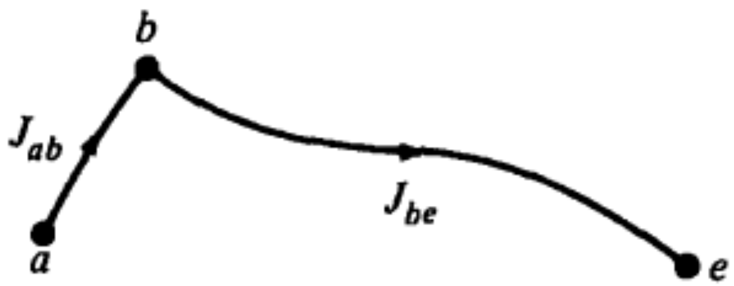
Other communities use different notation:

- Powell, W. B. *AI, OR and control theory: A Rosetta Stone for stochastic optimization*. Princeton University, 2012. `http://castlelab.princeton.edu/Papers/AIOR_July2012.pdf`

Principle of optimality

The key concept behind the dynamic programming approach is the principle of optimality

Consider deterministic case, and suppose the optimal path for a multi-stage decision-making problem is



- first decision yields segment a-b with cost J_{ab}
- remaining decisions yield segments b-e with cost J_{be}
- optimal cost is then $J^*_{ae} = J_{ab} + J_{be}$

Principle of optimality

Claim: If a-b-e is optimal path from a to e, then b-e is optimal path from b to e

Principle of optimality

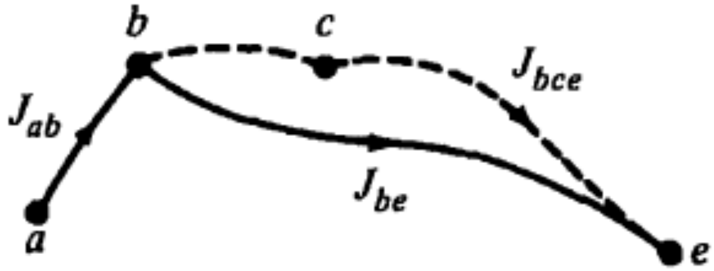
Claim: If a-b-e is optimal path from a to e, then b-e is optimal path from b to e

Proof: Suppose b-c-e is the optimal path from b to e. Then

$$J_{bce} < J_{be}$$

and

$$J_{ab} + J_{bce} < J_{ab} + J_{be} = J_{ae}^*$$



Principle of optimality

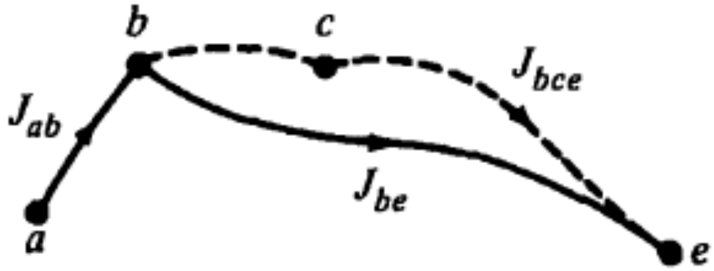
Claim: If a-b-e is optimal path from a to e, then b-e is optimal path from b to e

Proof: Suppose b-c-e is the optimal path from b to e. Then

$$J_{bce} < J_{be}$$

and

$$J_{ab} + J_{bce} < J_{ab} + J_{be} = J^*_{ae}$$



contradiction!

Principle of optimality

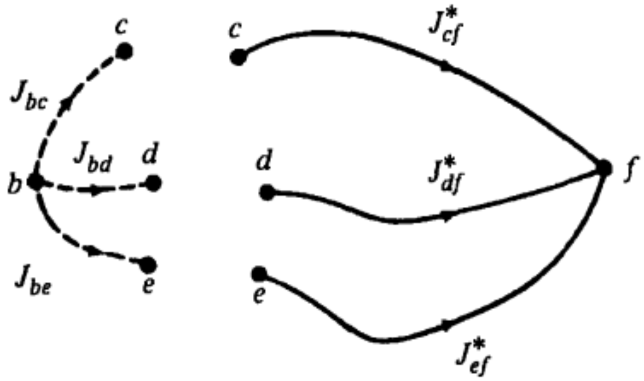
Principle of optimality (for discrete-time systems): Let $\mathbf{f}^* := \{\mathbf{f}_0^*, \mathbf{f}_1^*, \ldots, \mathbf{f}_{N-1}^*\}$ be an optimal policy. Assume state $\mathbf{x}_k$ is reachable. Consider the subproblem whereby we are at $\mathbf{x}_k$ at time k and we wish to minimize the cost-to-go from time k to time N. Then the truncated policy $\{\mathbf{f}_k^*, \mathbf{f}_{k+1}^*, \ldots, \mathbf{f}_{N-1}^*\}$ is optimal for the subproblem.

Principle of optimality

Principle of optimality (for discrete-time systems): Let $\mathbf{f}^* := \{\mathbf{f}_0^*, \mathbf{f}_1^*, \ldots, \mathbf{f}_{N-1}^*\}$ be an optimal policy. Assume state $\mathbf{x}_k$ is reachable. Consider the subproblem whereby we are at $\mathbf{x}_k$ at time k and we wish to minimize the cost-to-go from time k to time N. Then the truncated policy $\{\mathbf{f}_k^*, \mathbf{f}_{k+1}^*, \ldots, \mathbf{f}_{N-1}^*\}$ is optimal for the subproblem.

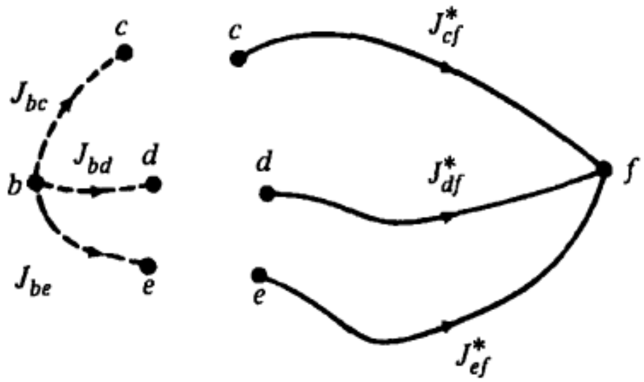
- tail policies optimal for tail subproblems
- notation: $\mathbf{f}_k^*(\mathbf{x}_k) = \mathbf{f}^*(\mathbf{x}_k, k)$

Applying the principle of optimality



Principle of optimality: if b-c is the initial segment of the optimal path from b to f, then c-f is the terminal segment of this path.

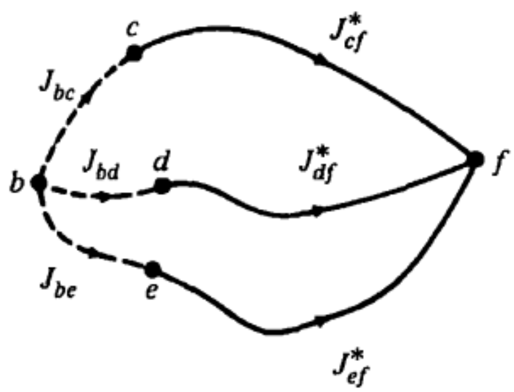
Applying the principle of optimality



Principle of optimality: if b-c is the initial segment of the optimal path from b to f, then c-f is the terminal segment of this path.

Hence, the optimal trajectory is found by comparing:

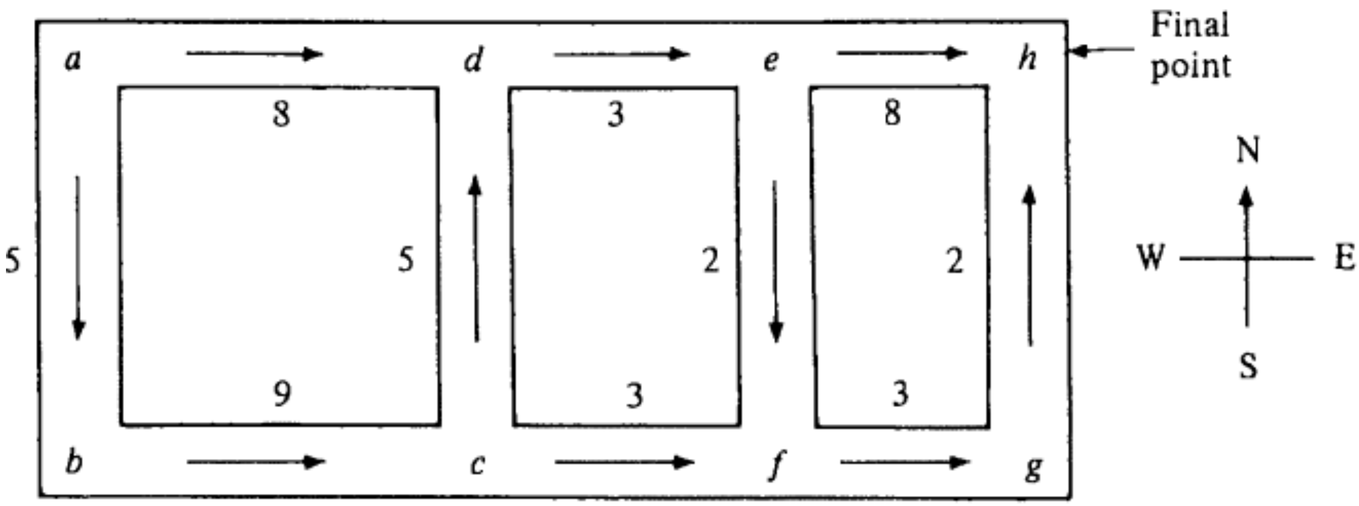
$$
\begin{aligned}
C_{bcf} &= J_{bc} + J^*_{cf} \\
C_{bdf} &= J_{bd} + J^*_{df} \\
C_{bef} &= J_{ec} + J^*_{ef}
\end{aligned}
$$



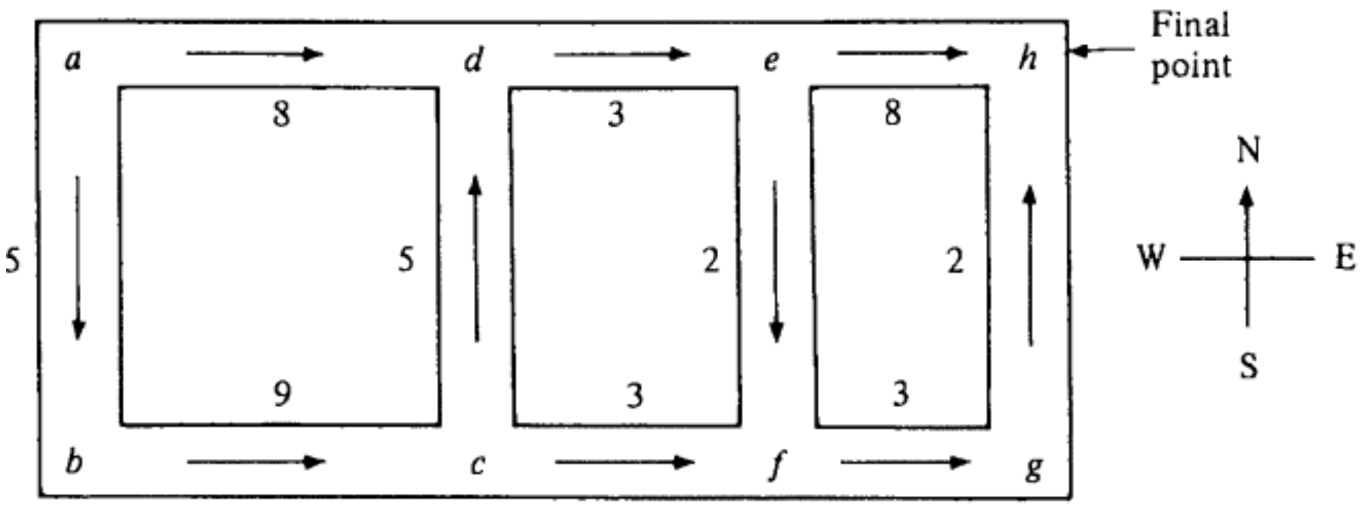
Applying the principle of optimality

- need only to compare the concatenations of immediate decisions and optimal decisions $\Rightarrow$ significant decrease in computation / possibilities
- in practice: carry out this procedure backward in time

Example



Example



Optimal path: $a \to d \to e \to f \to g \to h$
Optimal cost: 18

DP Algorithm

- Model: $\mathbf{x}_{k+1} = \mathbf{a}(\mathbf{x}_k, \mathbf{u}_k, k), \qquad \mathbf{u}_k \in U(\mathbf{x}_k)$
- Cost: $J_{\mathbf{f}}(\mathbf{x}_0) = h_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g(\mathbf{x}_k, \mathbf{f}_k(\mathbf{x}_k), k)$

DP Algorithm

- Model: $\mathbf{x}_{k+1} = \mathbf{a}(\mathbf{x}_k, \mathbf{u}_k, k), \qquad \mathbf{u}_k \in U(\mathbf{x}_k)$
- Cost: $J_{\mathbf{f}}(\mathbf{x}_0) = h_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g(\mathbf{x}_k, \mathbf{f}_k(\mathbf{x}_k), k)$

DP Algorithm: For every initial state $\mathbf{x}_0$, the optimal cost $J^*(\mathbf{x}_0)$ is equal to $J_0(\mathbf{x}_0)$, given by the last step of the following algorithm, which proceeds backward in time from stage $N-1$ to stage 0:

$$J_N(\mathbf{x}_N) = h_N(\mathbf{x}_N)$$
$$J_k(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k)} g_k(\mathbf{x}_k, \mathbf{u}_k, k) + J_{k+1}(\mathbf{a}(\mathbf{x}_k, \mathbf{u}_k, k)), \; k = 0, \ldots, N-1$$

Furthermore, if $\mathbf{u}_k^* = \mathbf{f}_k^*(\mathbf{x}_k)$ minimizes the right hand side of the above equation for each $\mathbf{x}_k$ and k, the policy $\{\mathbf{f}_0^*, \mathbf{f}_1^*, \ldots, \mathbf{f}_{N-1}^*\}$ is optimal

Back to the stochastic case

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy
- Consider tail subproblem

$$E\left\{g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right\}$$

and the tail policy $\{\mu_i^*, \ldots, \mu_{N-1}^*\}$

Back to the stochastic case

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy
- Consider tail subproblem

$$E\left\{g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right\}$$

  and the tail policy $\{\mu_i^*, \ldots, \mu_{N-1}^*\}$

- Principle of optimality: The tail policy is optimal for the tail subproblem

The DP Algorithm

Intuition:

- DP first solves ALL tail subroblems at the final stage
- At generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length

The DP Algorithm

Intuition:

- DP first solves ALL tail subroblems at the final stage
- At generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length

The DP algorithm:

- Start with
$$J_N(x_N) = g_N(x_N),$$
and go backwards using
$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(f(x_k, u_k, w_k)) \right\},$$
for $k = 0, 1, \ldots, N-1$
- Then $J^*(x_0) = J_0(x_0)$ and optimal policy is constructed by setting $\mu_k^*(x_k) = u_k^*$.

Example: Inventory Control Problem (1/2)

- Stock available $x_k \in \mathbb{N}$, inventory $u_k \in \mathbb{N}$, and demand $w_k \in \mathbb{N}$
- Dynamics: $x_{k+1} = \max(0, x_k + u_k - w_k)$
- Constraints: $x_k + u_k \leq 2$
- Probabilistic structure: $p(w_k = 0) = 0.1$, $p(w_k = 1) = 0.7$, and $p(w_k = 2) = 0.2$
- Cost

$$E\left\{\underbrace{0}_{g_3(x_3)} + \sum_{k=0}^{2} \underbrace{\left(u_k + (x_k + u_k - w_k)^2\right)}_{g(x_k, u_k, w_k)}\right\}$$

Example: Inventory Control Problem (2/2)

- Algorithm takes form

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

for $k = 0, 1, 2$

Example: Inventory Control Problem (2/2)

- Algorithm takes form

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

for $k = 0, 1, 2$

- For example

$$J_2(0) = \min_{u_2 = 0,1,2} E_{w_2} \left\{ u_2 + (u_2 - w_2)^2 \right\} = \min_{u_2 = 0,1,2} \left[u_2 + 0.1(u_2)^2 + 0.7(u_2 - 1)^2 + 0.2(u_2 - 2)^2 \right]$$

which yields $J_2(0) = 1.3$, and $\mu_2^*(0) = 1$

Example: Inventory Control Problem (2/2)

- Algorithm takes form

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

for $k = 0, 1, 2$

- For example

$$J_2(0) = \min_{u_2 = 0,1,2} E_{w_2} \left\{ u_2 + (u_2 - w_2)^2 \right\} = \min_{u_2 = 0,1,2} \left[u_2 + 0.1(u_2)^2 + 0.7(u_2 - 1)^2 + 0.2(u_2 - 2)^2 \right]$$

which yields $J_2(0) = 1.3$, and $\mu_2^*(0) = 1$

- Final solution $J_0(0) = 3.7$, $J_0(1) = 2.7$, and $J_0(2) = 2.818$

Difficulties of DP

- Curse of dimensionality:
  - Exponential growth of the computational and storage requirements
  - Intractability of imperfect state information problems
- Curse of modeling: if "system stochastics" are complex, it is difficult to obtain expressions for the transition probabilities
- Curse of time
  - The data of the problem to be solved is given with little advance notice
  - The problem data may change as the system is controlled—need for on-line replanning

Solution: Approximate DP

- Certainty Equivalent Control
- Cost-to-Go Approximation
- Other Approaches (e.g., approximation in policy space)

Certainty Equivalent Control

- Idea: Replace the stochastic problem with a deterministic one
- At each time "k," the future uncertain quantities are fixed at some "typical" values
- Online implementation
  1. Fix the w_i, $i \geq k$, at some $\bar{w}_i$ and solve deterministic problem

$$\min g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \bar{w}_i)$$

     where $x_{i+1} = f_i(x_i, u_i, \bar{w}_i)$
  2. Use as control $\bar{\mu}_k(x_k)$ the first element of optimal control sequence and move to step $k+1$

Certainty Equivalent Control

- Idea: Replace the stochastic problem with a deterministic one
- At each time "k," the future uncertain quantities are fixed at some "typical" values
- Online implementation
  1. Fix the w_i, $i \geq k$, at some $\bar{w}_i$ and solve deterministic problem
  $$\min g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \bar{w}_i)$$
  where $x_{i+1} = f_i(x_i, u_i, \bar{w}_i)$
  2. Use as control $\bar{\mu}_k(x_k)$ the first element of optimal control sequence and move to step $k+1$

Cost-to-Go Approximation (CGA)

- Idea: Truncate time horizon and approximate cost-to-go

- One-step lookahead policy: at each k and state x_k, use control $\bar{\mu}_k(x_k)$ that

$$\min_{u_k \in U_k(x_k)} E\left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\},$$

  - $\tilde{J}_N = g_N$
  - $\tilde{J}_{k+1}$: approximation to true-cost-to-go J_{k+1}

- Analogously, two-step lookahead policy: all of the above and

$$\begin{aligned}\tilde{J}_{k+1}(x_{k+1}) = \min_{u_{k+1} \in U_{k+1}(x_{k+1})} E\big\{ &g_{k+1}(x_{k+1}, u_{k+1}, w_{k+1}) \\ &+ \tilde{J}_{k+2}(f_{k+1}(x_{k+1}, u_{k+1}, w_{k+1})) \big\}\end{aligned}$$

CGA—Computational Aspects

- If $\tilde{J}_{k+1}$ is readily available and minimization not too hard, this approach is implementable on-line
- Choice of approximating functions $\tilde{J}_k$ is critical
  1. Problem Approximation: approximate by considering simpler problem
  2. Parametric Cost-to-Go Approximation: approximate cost-to-go function with function of suitable parametric form (parameters tuned by some scheme $\rightarrow$ neuro-dynamic programming)
  3. Rollout Approach: approximate cost-to-go with cost of some suboptimal policy

CGA—Problem Approximation

- Many problem-dependent possibilities
  - Replace uncertain quantities by nominal values (in the spirit of CEC)
  - Simplify difficult constraints or dynamics
  - Decouple subsystems
  - Aggregate states

CGA—Parametric Approximation

- Use a cost-to-go approximation from a parametric class $\tilde{J}(x, r)$ where x is the current state and $r = (r_1, \ldots, r_m)$ is a vector of "tunable" weights
- Two key aspects
  - Choice of parametric class $\tilde{J}(x, r)$
    - Example: feature extraction method

$$\tilde{J}(x, r) = \sum_{i=1}^{m} r_i \, y_i(x),$$

      where the y_i's are *features*
  - Algorithm for tuning the weights (possibly, simulation-based)

CGA—Rollout Approach

- $\tilde{J}_k$ is the cost-to-go of some heuristic policy (called the *base policy*)
- To compute rollout control, one need for all u_k

$$Q_k(x_k, u_k) := E\left\{g_k(x_k, u_k, w_k) + H_{k+1}(f_k(x_k, u_k, w_k))\right\},$$

  where H_{k+1} is the value of the cost-to-go for the base policy

  - Q-factors can be evaluated via Monte-Carlo simulation
  - Q-factors can be approximated, e.g., by using a CEC approach

- Model predictive control (MPC) can be viewed as a special case of rollout algorithms (AA 203)

Other ADP Approaches

- Minimize the DP equation error
- Direct approximation of control policies
- Approximation in policy space

References

- Bertsekas, D. P. *Dynamic programming and optimal control*. Volumes 1 & 2. Athena Scientific, 2005.
- Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 1998.